UAI 8-31900 FR I
VOL. II

(NASA-CR-150[illegible]4) NASTRAN HYDROELASTIC MODAL STUDIES. VOLUME 2: PROGRAMMER DOCUMENTATION Final Report (Universal Analytics, Inc.) 137 p HC A07/MF A01
CSCL 20D G3/34

N79-25352

Unclas
26862

UNIVERSAL ANALYTICS I[illegible]

Los Angeles

FINAL REPORT
NASTRAN HYDROELASTIC MODAL STUDIES

VOLUME II

PROGRAMMER DOCUMENTATION

Prepared for

National Aeronautics and Space Administration
Marshall Space Flight Center

Under Contract No. NAS8-31900

May 31, 1977

Prepared by

UNIVERSAL ANALYTICS, INC.
7740 West Manchester Boulevard
Playa Del Rey, California 90291

(213) 822-4422

FOREWORD

This volume contains the programming documentation for the NASTRAN updates and the MESHGEN program for the NASTRAN Hydroelastic System. Volume I contains the Theory and Results. Volume III contains the User Manual updates.

VOLUME II - TABLE OF CONTENTS

4.0 NASTRAN PROGRAM MANUAL UPDATES

The operational steps, data descriptions, and program code for the new NASTRAN hydroelastic analysis system are described in this section. The format is similar to that of the NASTRAN Programmer's Manual for the module and subroutine descriptions. The overall flow of the system is described, followed by the descriptions of the individual modules and their subroutines.

4.1 OVERALL PROGRAM FLOW

From a programmer's viewpoint the NASTRAN hydroelastic normal modes analysis is a straight forward extension of Rigid Format 3, the structural normal modes analysis. The overall flow of the problem for a direct formulation, shown in Figure 1, describes the key steps and data blocks used in the DMAP program. Because of the special characteristics of the fluid degrees of freedom, the USET definitions in NASTRAN have been modified. These components describe the size and characteristics of the matrices and vectors.

The names of these new degree-of-freedom sets are defined below:

u_g: All structure and fluid components

u_x: Structure components

u_y: Fluid components

u_{fr}: Free surface degrees of freedom

$\bar{u}_a$: Reduced structure components

u_a: Reduced structure plus free surface displacements = solution set

In the DMAP the names of matrices and vectors generally correspond to these sets. For instance, AXY is a matrix with rows defined by structure points (x) and columns corresponding to fluid points (y).

The steps described in Figure 1 are:

Step 1: The normal steps used in Rigid Format 3 are used to build the stiffness (KGG) and mass (MGG) matrices for both structure and fluid. The matrix terms for the fluid points are actually pressure/flow functions rather than the normal displacement/force functions. The

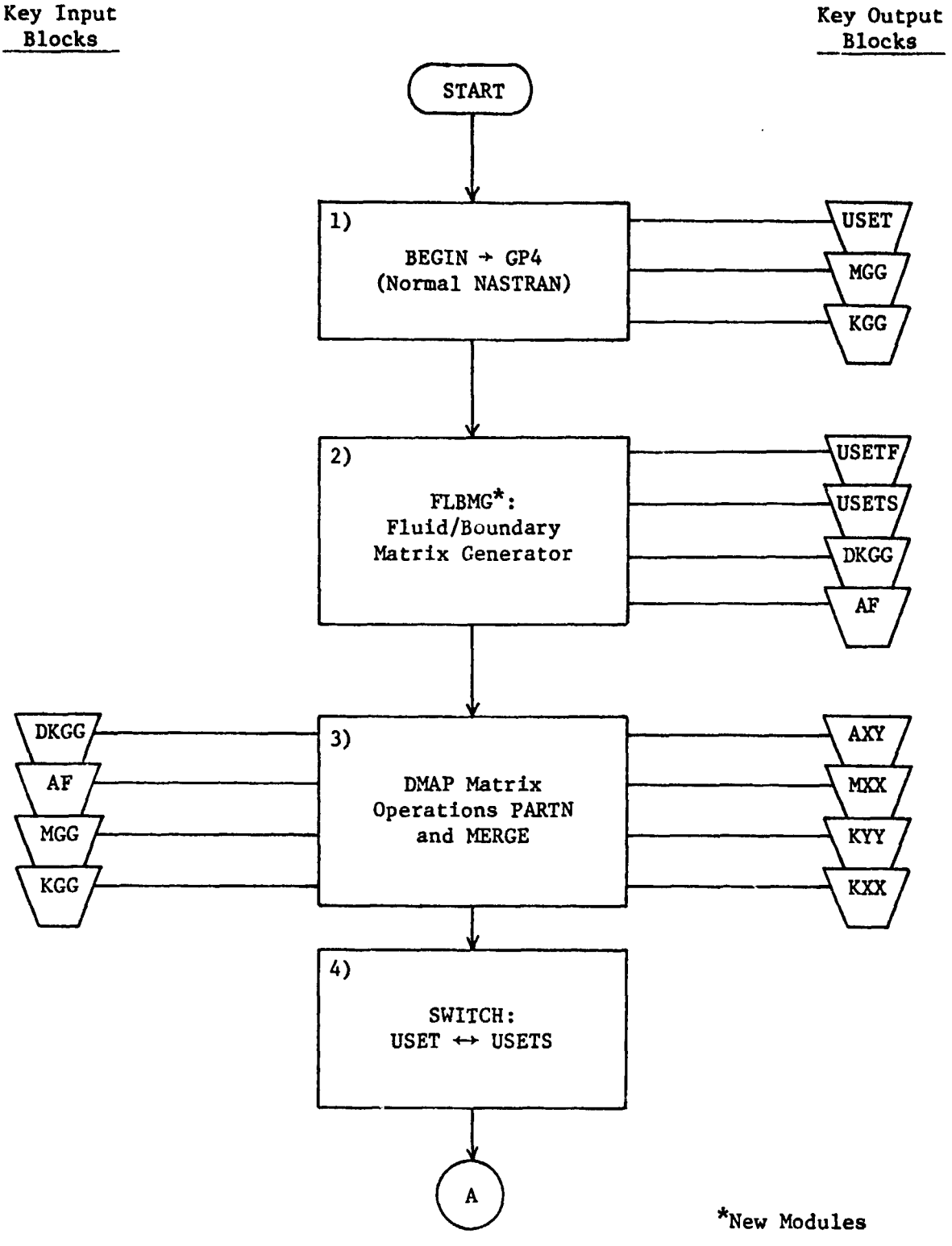


FIGURE 1. NASTRAN HYDROELASTIC DMAP FLOW (Direct Assembly)

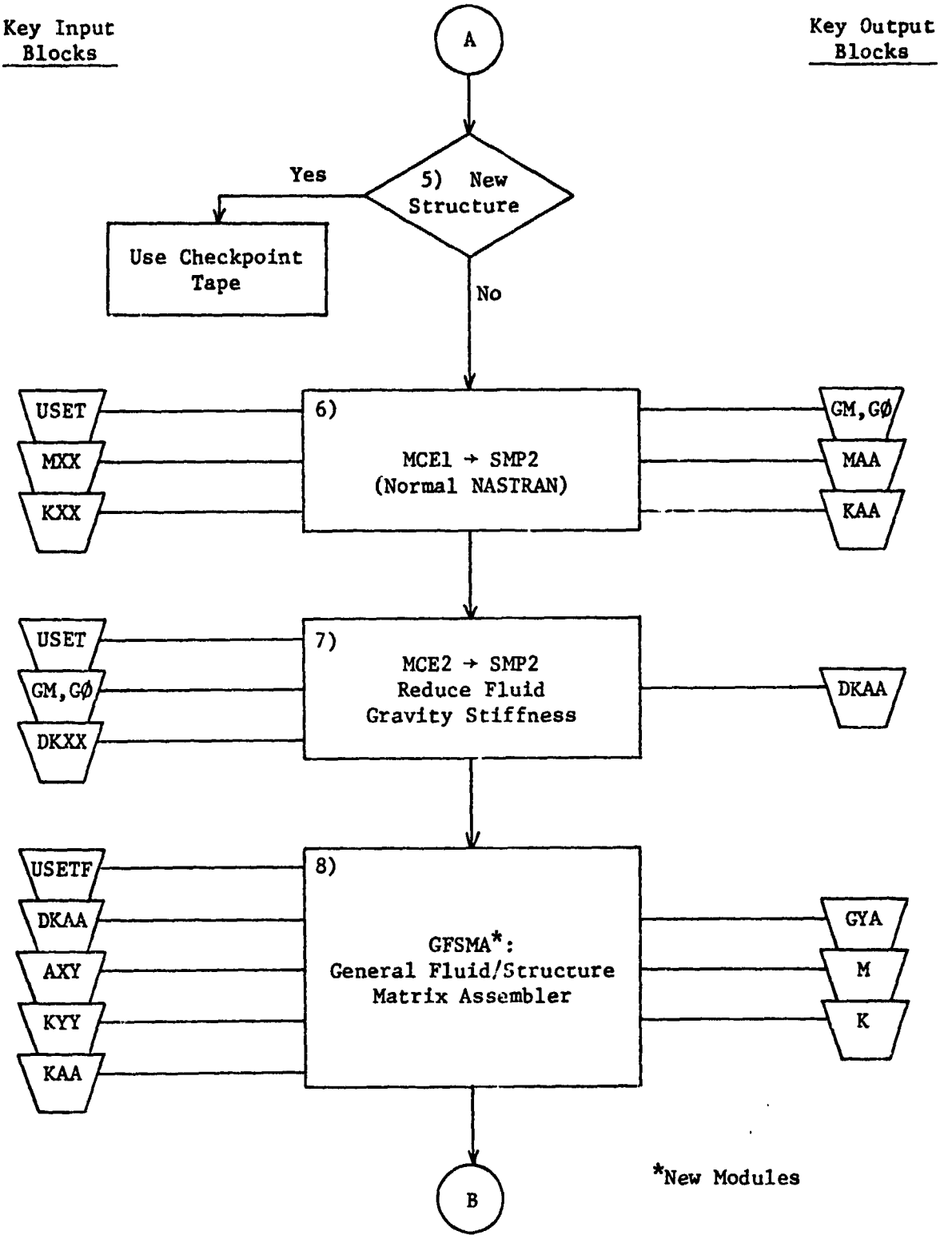


FIGURE 1. NASTRAN HYDROELASTIC DMAP FLOW (Cont'd)

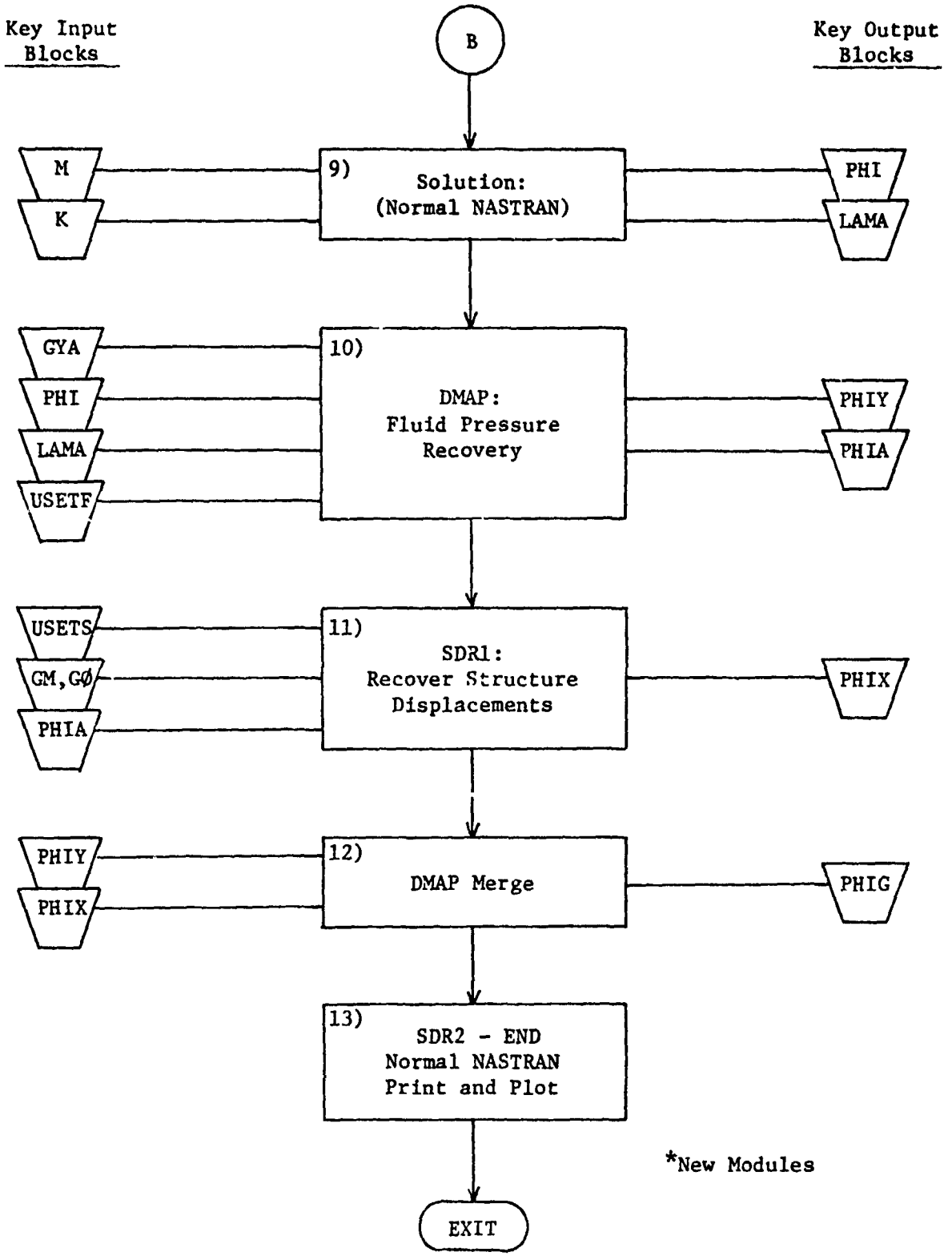


FIGURE 1. NASTRAN HYDROELASTIC DMAP FLOW (Cont'd)

USET data block, generated by module GP4, describes all degrees of freedom.

Step 2: The FLBMG (Fluid/Boundary Matrix Generator) Module generates the area factor matrix, AF, and the additional stiffness due to gravity, DKGG. It also produces two new USET data blocks. USETF contains the fluid point flags and USETS describes only structure points.

Step 3: The following DMAP partitioning operations are coded in the system:

a) Partition g → x + y

$$[K_{gg}] \Rightarrow \left[\begin{array}{c|c} K_{xx} & 0 \\ \hline 0 & K_{yy} \end{array}\right]$$

$$[M_{gg}] \Rightarrow \left[\begin{array}{c|c} M_{xx} & 0 \\ \hline 0 & 0 \end{array}\right]$$

$$[R_y] \Rightarrow [R_x \mid 0]$$

$$[AF] \Rightarrow \left[\begin{array}{c|c} 0 & A_{xy} \\ \hline 0 & A_{yy} \end{array}\right]$$

$$[DK_{gg}] \Rightarrow \left[\begin{array}{c|c} DK_{xx} & 0 \\ \hline 0 & DK_{yy} \end{array}\right]$$

b) Partition y → fr + i

$$[DK_{yy}] \Rightarrow \left[\begin{array}{c|c} DK_{frfr} & 0 \\ \hline 0 & 0 \end{array}\right]$$

$$[A_{yy}] \Rightarrow \left[\begin{array}{c} A_{fry} \\ \hline 0 \end{array}\right]$$

Step 4: The SWITCH module is used to replace the USET data block with the new block which describes only structure points. The normal structure processing modules will use this data block and operate only on the subsets of the u_x set.

Steps 5 & 6: For non-restart problems, the structure stiffness and mass matrices are constrained and reduced in the usual manner. For restart problems with no structure changes, these operations are bypassed by the use of a PARAM bulk data card.

Step 7: For fluid problems with gravity effects, the incremental stiffness matrix, DKXX, is constrained and reduced to correspond to the reduced structure points.

Step 8: Module GFSMA (General Fluid/Structure Matrix Assembler) performs the operations to merge the structure and fluid matrices into the solution matrices (see Section 4.4 for the detailed steps). The output matrices correspond to the reduced structural displacements and the free surface displacements.

Step 9: The NASTRAN solution modules are executed at this stage. For normal modes, the READ (Real Eigenvalue Analysis - Displacement method) module is used. However, the capability exists to restart in a dynamics analysis rigid format.

Step 10: MPYAD is used to recover fluid point pressures and free surface point displacements:

$$[G_y][\phi] = [\phi_i]$$

The structure and free surface displacements are separated, d → a + fr.

$$[\phi] \Rightarrow \begin{bmatrix} \phi_a \\ \hline \phi_{fr} \end{bmatrix}$$

The fluid pressures and free surface displacements are merged:

$$\begin{bmatrix} \phi_{fr} \\ \hline \phi_i \end{bmatrix} \Rightarrow [\phi_y]$$

Steps 11 - 13: The dependent displacements for the structure points are recovered with Module SDR1, the fluid displacements and pressures are merged with the displacement vectors, and the normal NASTRAN

output processing is executed. The displacements of fluid free surface points are printed and plotted. The pressures on the interior fluid points may be printed. All structure displacement, force, stress, etc. data is processed with the standard routines.

4.2 FUNCTIONAL MODULE FLBMG (FLUID BOUNDARY MATRIX GENERATOR)

Entry Point: FLBMG

Purpose

To assemble the hydroelastic area factor and gravity stiffness matrices. In addition, the hydroelastic USET vector is constructed for defining the various structural and fluid sets.

DMAP Calling Sequence

```
FLBMG   GEØM2,ECT,BGPDT,SIL,MPT,GEØM3,CSTM,USET,EQEXIN/USETF,USETS,AF,
        DKGG/S,N,NØGRAV/S,N,NØFREE/S,N,TILT $
```

Input Data Blocks

GEØM2 - Element Connection Data
ECT - Element Connection Table
BGPDT - Basic Grid Point Definition Table
SIL - Scalar Index List
MPT - Material Properties Table
GEØM3 - Gravity Load Data
CSTM - Coordinate System Transformation Matrices
USET - Displacement Set Definition Table
EQEXIN - Equivalence between external and internal grid points

Notes

1. The CSTM may be purged.
2. GEØM3 may be purged only if no gravity effects are computed.

Output Data Blocks

USETF - Set Definition Table, fluid and structural points
USETS - Set Definition Table, structural points only
AF - Fluid Area Factor Matrix
DKGG - Gravity Stiffness Matrix

Note

1. DKGG will be purged if no gravity effect is computed.

Parameters

NØGRAV - Input - integer - no default. Flag which specifies whether gravity effects are to be computed (-1 ignores gravity).

NØFREE - Output - integer - no default. Flag which specifies whether a fluid free surface exists (-1 implies no free surface).

TILT - Output - complex - default = (1.0,0.0). Components of the normal vector to the free surface. Used as input to SDR2 for plotting.

Method

Subroutine FLBMG is a small driver which calls the following routines to perform the various phases of matrix generation and assembly.

1. FLBELM processes the fluid/structural boundary elements and the fluid free surface elements. In addition, vectors describing the connectivity of fluid and structural elements are created for use when allocating core during matrix assembly.
2. FLBSET produces the two new fluid USET vectors for use in the hydroelastic analysis.
3. FLBEMG calculates the individual element area factor and element gravity stiffness matrices.
4. FLBEMA assembles the element matrices into full size output matrices.

Subroutines

The utility subroutines PRETRD and PREMAT are called by FLBEMG so that lower level subroutines may call entry points TRANSD and MAT to obtain coordinate system transformation matrices (CSTM) data and material properties data, respectively. The utility routines GMMATD, DCRØSS, and DNØRM are also used for basic in-core matrix computations.

For purposes of communication between subroutines of the FLBMG module, the following common blocks are used.

Block	Variables in Order
/FLBPTR/	ERROR - Fatal error flag
	ICØRE - Next available word of core
	LCØRE - Length of open core
	IBGPDT - Start of BGPDT data

	NBGPDT	- Length of data
	ISIL	- Start of SIL data
	NSIL	- Length of data
	IGRAV	- Start of GRAV data
	NGRAV	- Length of data
	IGRID	- Start of Grid Point Connectivity Table
	NGRID	- Length of table
	IBUF1	- GINØ buffer 1
	IBUF2	- GINØ buffer 2
	IBUF3	- GINØ buffer 3
	IBUF4	- GINØ buffer 4
	IBUF5	- GINØ buffer 5
/FLBZZ1/	Z	- Open core for the first three phases of matrix generation.
/FLBZZ2/	Z	- Open core for the last phase, or matrix assembly.
/FLBFIL/	GEØM2	GINØ file numbers for the various input, output and scratch files.
	ECT	
	BGPDT	
	SIL	
	MPT	
	GEØM3	
	CSTM	
	USET	
	EZEXIN	
	USETF	
	USETS	
	AF	
	DKGG	
	FBELM	
	FRELM	
	CØNECT	
	AFMAT	
	AFDICT	
	KGMAT	
	KGDICT	

Subroutine Name: FLBELM

Entry Point: FLBELM

Purpose: To process the fluid elements constituting the fluid/structure boundary and to determine connectivity between these elements.

Calling Sequence: CALL FLBELM

Method: The BGPDT data describing grid points is opened and read into open core. GEØM2 is then opened and the record containing CFLSTR bulk data cards is located. The data is then read and stored below the BGPDT data in the Element Boundary Table as follows:

Word No.	Description
1	Fluid element ID
2	Structural element ID
3-6	(zero)
7	Gravity ID

The ECT data block is then read and each 2-D element type is processed. As each element is processed, if the structural element is connected to any fluid element, the grid points of the structural element are inserted into the four zero words in the element boundary table record. (If the structural element is triangular, only three grids are found.) The CFREE cards are then read from GEØM2 and stored in a similar seven-word record in core.

Word No.	Description
1	Fluid element ID
2	-1
3	Element FACE ID
4-6	zero
7	Gravity ID

The ECT is then read again and only fluid elements are processed. For each element the following steps are performed.

If the fluid element is on the free surface (the structural element ID will be negative):

1. Subroutine FLFACE is called to locate the 3 or 4 fluid grid points on the desired face which forms the free surface.

2. A 7-word record is written on file FRELM which contains the fluid element ID, material ID, gravity ID, and the fluid grid points defining the desired face.

If the fluid element is connected to a structural element:

1. Subroutine FLFACE is called to locate the 3 or 4 fluid grid points on the face which forms the boundary.

2. A 12-word record is written on file FBELM which is similar to FRELM except the structural element ID and grid points are also included.

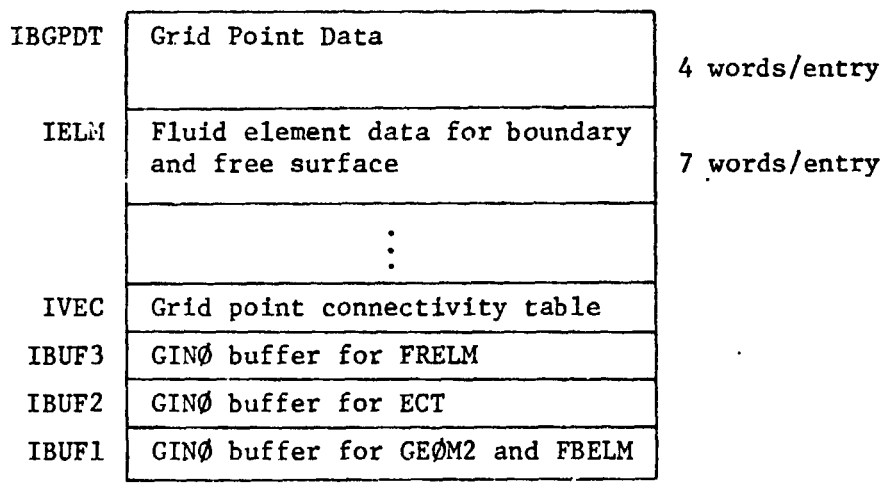
As the above steps are being performed a table of grid point connectivity is also created. This table provides three pieces of information:

1. The maximum number of structural grid points connected to each fluid point on the boundary.

2. The maximum number of structural grid points connected to each structural point on the boundary.

3. The maximum number of fluid points connected to each fluid point on the free surface.

The open core layout for FLBELM is as follows:



Subprogram Name: FLFACE

Entry Point: FLFACE

Purpose: To locate the fluid grid points which describe the face of a fluid element. The particular face may be specified by either a face number or the grids of a structural element which form a boundary with the face.

Calling Sequence: CALL FLFACE (TYPE,ECT,EBT,ELT)

TYPE - The element type

ECT - The ECT record for the desired fluid element

ELT - The element boundary table entry for the fluid element. The 7-word block, as described in the previous section, contains either the fluid face number or the structural grid ID's.

GRID - Fluid grid points located (3 or 4)

Method: If the fact ID is presented:

The corresponding fluid grid points are obtained directly. The face numbering conventions are described in the User's Manual under the bulk data description for each fluid element type.

If the structural grid points are provided:

1. Find normal and centroid of structural element face

$$\vec{R}_{12} = \vec{R}_2 - \vec{R}_1$$

$$\vec{R}_{13} = \vec{R}_3 - \vec{R}_1$$

$$\vec{K}_5 = \frac{\vec{R}_{12} \times \vec{R}_{13}}{|\vec{R}_{12} \times \vec{R}_{13}|}$$

$$\vec{R}_c = \frac{1}{3}(\vec{R}_1 + \vec{R}_2 + \vec{R}_3) \quad \text{for triangle element}$$

$$\vec{R}_c = \frac{1}{4}(\vec{R}_1 + \vec{R}_2 + \vec{R}_3 + \vec{R}_4) \quad \text{for quad element}$$

where R_i are coordinates for structural grid points obtained from the BGPDT.

2. For each face of the fluid element, find the angle between the fluid face and structure face. If this angle is less than 30°, also find the distance from the centroid of the structure element normal to the face of the fluid element.

$$\vec{r}_{12} = \vec{r}_2 - \vec{r}_1$$

$$\vec{r}_{13} = \vec{r}_3 - \vec{r}_1$$

$$\vec{K}_f = \frac{\vec{r}_{12} \times \vec{r}_{13}}{|\vec{r}_{12} \times \vec{r}_{13}|}$$

IF $|\vec{K}_f \cdot \vec{K}_s| < 0.866$ THEN

$$\Delta h = |\vec{K}_f \cdot (\vec{R}_c - \vec{r}_1)|$$

where r_i are coordinates for fluid grid points obtained from the BGPDT.

3. The face choosen has the smallest Δh value with the angle between faces less than 30°.

Subprogram Name: FLBSET

Entry Point: FLBSET

Purpose: To construct the hydroelastic USET vectors.

Calling Sequence: CALL FLBSET

Method: The SIL list is read into core beneath the BGPDT and the USET vector after that. The list of grid point connectivity is then written to the CONECT file in two files, one file for each matrix to be assembled later. The files consist of three-word records as follows:

Word No.	Description
1	SIL number
2	Maximum grid points connected to this SIL
3	Maximum SIL's connected

The USET table is then read into core. A list of free surface grid points is then constructed by examining each entry in the grid point connectivity table. The BGPDT is then scanned and the following USET bits are set:

If fluid point:

Set Y bit

Search list of free surface grids; if present, set fr, Z, and a bit, otherwise point is interior fluid point and set i bit.

If structural point:

Set X and Z bit

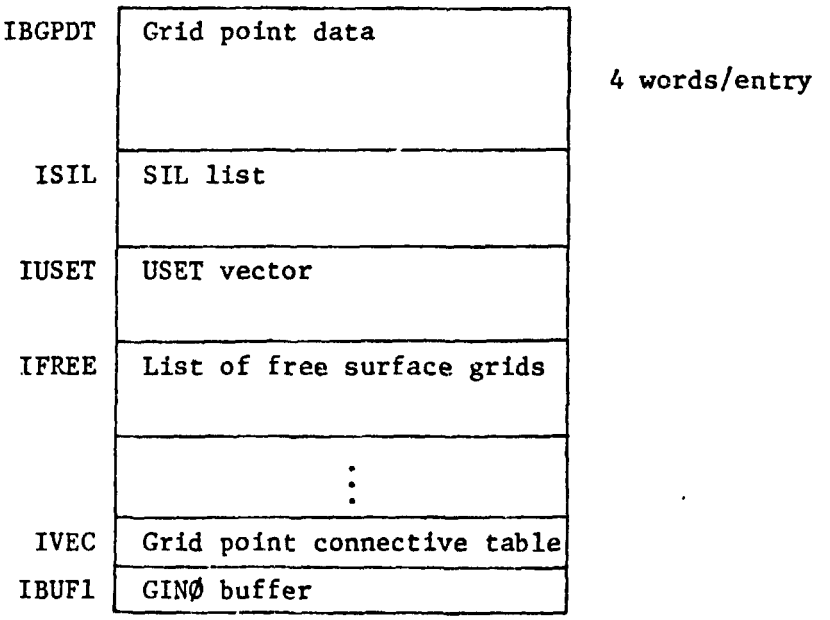
If a bit is set, set $\bar{a}$

Hydroelastic USET bit positions are as follows:

Set	Bit Position	Description
U_a	25	$\bar{a}$ + fr
U_x	9	Structure only
U_y	8	Fluid only
U_{fr}	7	Free surface
U_z	6	x + fr
$U_{\bar{a}}$	5	$\bar{a}$ bits (structure only)
U_i	4	Interior fluid point

The new USET vector is then written to the USETf file. In addition, only the structure points, x bit set, are written to the USETS file.

The open core layout for FLBSET is as follows:



Subprogram Name: FLBPRT

Entry Point: FLBPRT

Purpose:

1. Prints, at user request via DIAG 23, a list of degrees of freedom. For each degree of freedom, an indication is made identifying the sets to which it belongs.

2. Prints, at user request via DIAG 24, the contents of selected displacement sets. For each set, a list of all degrees of freedom belonging to the set given.

Calling Sequence: CALL FLBPRT (IUSET,ICØRE,IBUF)

IUSET - Open core location of USETF table
ICORE - Start of open core for storage of EQEXIN
IBUF - GINØ buffer location

Method: The DIAG flags are tested and local variables set. Table EQEXIN is then read into open core and sorted. If DIAG 23 is set, Table USETF (already in open core) is examined and the external degree of freedom is extracted from EQEXIN and printed along with the set indications. If DIAG 24 is set, the transpose process takes place.

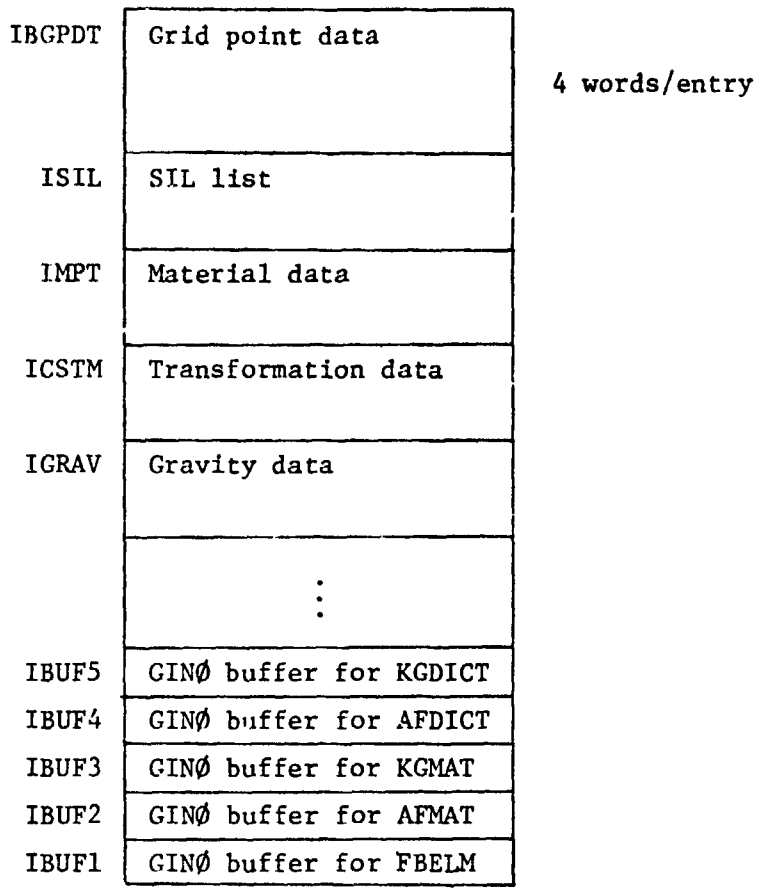
Subprogram Name: FLBEMG

Entry Point: FLBEMG

Purpose: To calculate element gravity stiffness and area factor matrices. These element matrices, along with dictionary entries to describe them, are written to files for use in the matrix assembly phase.

Calling Sequence: CALL FLBEMG

Method: The material property data is read into core below the SIL list using utility PREMAT. PRETRD is then called to set up coordinate transformation data in core. GEØM3 is then opened and GRAV bulk data cards are located and read into open core. The open core layout for FLBEMG will then be:



File FBELM is then read and, for each element, subroutine BØUND is called to calculate the area factor matrix and gravity stiffness matrix for the element. The gravity stiffness calculations will be performed only if gravity loads are present. These matrices are then written to files AFMAT and KGMAT respectively. An AFMAT entry contains several 3 x 1 matrix partitions as follows:

Word No.	Description
1-3	fluid SILS
4-6	structure SILS
for triangle faces	
7-60	9 x 3 matrix
for quad faces	
7-102	12 x 4 matrix

The KGMAT file is written in a similar manner except is consists of 3 x 3 partitions. Dictionary entries are then written to the AFDICT and

KGDICT files to describe each column of the above matrix partitions. The file position is also recorded so the matrix partitions may be accessed in a random sequence during assembly. A AFDICT entry is as follows:

Word No.	Description
1	Column SIL number
2	File position

After the FBELM file is exhausted, a similar procedure is performed with the FRELM file. The major difference is that subroutine FLFREE is called to compute the free surface effects on these elements.

Subprogram Name: FLFREE

Entry Point: FLFREE

Purpose: To compute the area factor and gravitational stiffness matrices for a fluid element on the free surface.

Calling Sequence: CALL FLFREE (FRREC,AFE,NAFE,KGE,NKGE)

FRREC - FRELM record for a fluid element

AFE - Element area factor matrix. Maximum size is 4 x 4 double precision words

NAFE - Number of words in the AFE matrix

KGE - Element gravity stiffness matrix. Maximum size is 4 x 4 double precision words

NKGE - Number of words in the KGE matrix

Method:

1. If the fluid face is quadratic, it is divided into four overlapping triangles for steps 2 through 4.
2. The area, A, of the fluid triangle is computed.
3. The area factor terms for this triangle are computed and inserted into the full size element matrix.

$$[A_{ij}] = \begin{cases} A/12 & i \neq j \\ A/6 & i = j \end{cases}$$

for

$i = 1,2,3$

$j = 1,2,3$

4. If gravity loads are requested, the additional stiffness terms are computed and inserted into the element matrix.

$$[K_{ij}] = \rho g[A_{ij}]$$

for

$i = 1,2,3$

$j = 1,2,3$

where

ρ = fluid density

g = gravitation constant

5. If the fluid face is quadratic, the area factor and gravity matrices are divided by two to account for the overlapping triangles.

Subprogram Name: FLBEMA

Entry Point: FLBEMA

Purpose: To assemble the element matrices into either the AF or DKGG matrix.

Calling Sequence: CALL FLBEMA (TYPE)

TYPE - 1 for AF matrix
2 for DKGG matrix

Method: One G-size vector is allocated at the top of open core. This vector will be the column list, where each entry points to the open core location where data for that column is held. The column vector is then zeroed and the CØNECT file is opened and positioned for the type of matrix being built. As each entry from the CØNECT file is read, the necessary core is allocated to hold the terms for that column. A pointer to this core is inserted into the column pointer vector. The amount of core required for each column is:

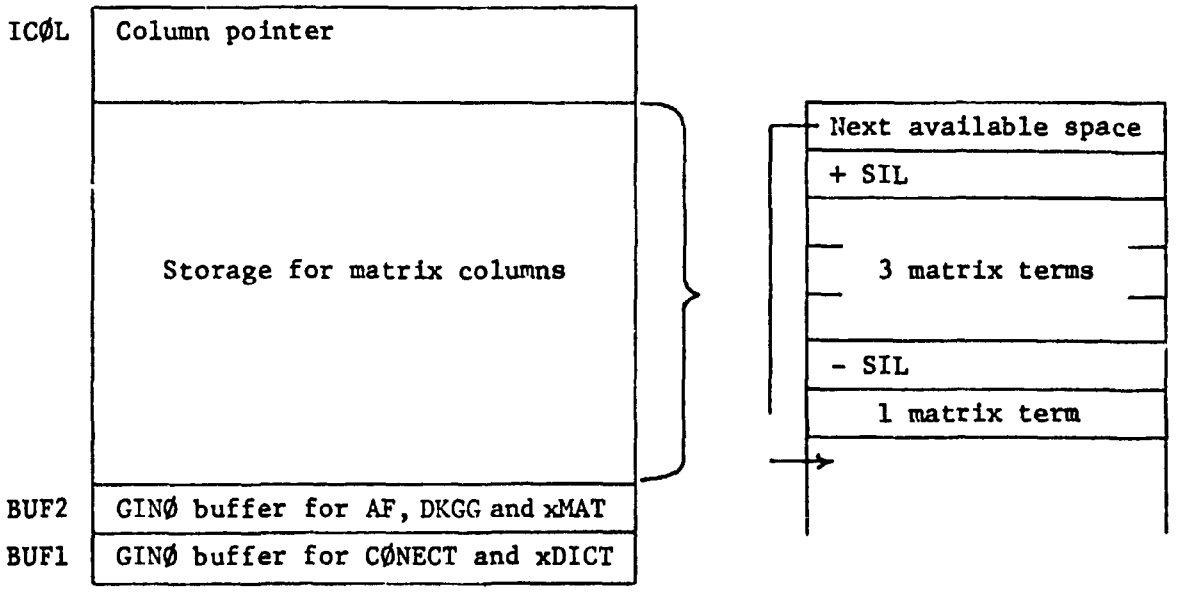
1 + MAX GRIDS + 2 * MAX SILS

This area will then hold one SIL value and up to 3 double precision matrix terms for each grid point entry in the column.

This process is continued until the CØNECT file is exhausted or insuffient core remains to held the next column. The appropriate dictionary file is then read and for each entry it is determined if the desired column is in core for this pass. If it is, the matrix terms are obtained from the matrix term file, xMAT, and stored in core. When all dictionary entries are processed, each column in core is written to the output matrix with subroutine PAKCØL. Note that only the columns with nonzero terms are stored in core, making it necessary to pack out null columns for the others.

If any non-zero columns remain, i.e., entries remain in the CØNECT file, core is reallocated and additional passes are made through dictionary and matrix term files to build these columns.

The open core layout for the assembly phase is as follows:



Subroutine Name: PAKCØL

Entry Point: PAKCØL

Purpose: To build the single matrix columns and pack it out to the output matrix.

Calling Sequence: CALL PAKCØL (REC,LREC)

REC - In-core record containing the SIL's and matrix terms for the column

LREC - Length of this record

Method:

1. The entries for each SIL (one SIL and up to three matrix terms) are sorted on the absolute value of the SIL.
2. If any duplicate SIL's exist, their terms are added together.
3. The individual matrix terms are then packed to the file using ZBLPKI.

Subroutine Name: BØUND

Entry Point: BØUND

Purpose: To obtain area factor and stiffness matrices as function of area common between specified fluid and structural element faces.

Common Sequence: CALL BØUND(FBREC,AFE,NAFE,KGE,NKGE)

FBREC(12) - Fluid boundary record:
- (1) Fluid element ID
- (2) Structural element ID
- (3-6) (internal) Grid nos. for structural element face
- (7) Gravity ID
- (8) Material ID
- (9-12) (internal) Grid nos. for fluid element face

AFE(48) - Area factor matric storage

NAFE - Number of (single precision) storage locations in AFE returned by subroutine BØUND.

KGE - Stiffness matrix storage.

NKGE - Number of (single precision) storage locations in KGE returned by subroutine BØUND.

Method: After transforming coordinates of both element faces into system in which x/y coordinates of point 1 and y coordinates of point 2 of the fluid element are zero, the area common to both faces when projected on the x,y plane is obtained by determining the coordinates bounding this polygon of common area. The load distribution factors and the area factors are then calculated by isoparametric interpolation. After any required gravitational stiffness terms have been added to the stiffness matrix, both the area factor (AFE) and stiffness (KGE) matrices are transformed to global coordinate system (if required) and the stiffness matrix is rearranged to column storage to conform to NASTRAN requirements.

Design Requirements/Limitations: Neither element (face) may have more than four edges. If there is no common area, User Warning Message 8014 is

printed. If an undefined gravity ID is referenced, User Fatal Message 8013 is printed. When element geometry is such that isoparametric interpolation fails, User Fatal Message 8005 is printed.

Subroutine PTINTR (PoinT of INTeRsection)

Entry Point: None

Purpose: Returns (double precision) xy coordinates of point of intersection (if any) of two lines.

Calling Sequence: CALL PTINTR(A,AA,B,BB,S,K,EPS)

A(2),AA(2) - xy coordinates of end points of line segments A and AA,
B(2),BB(2) and B and BB (real, double precision)

S(2) - xy coordinates (returned) of point of intersection of line segments AAA and BBB; S = (0,0) if no point of intersection (real, double precision).

K - Condition flag (integer) returned (single precision):
= 1, point of intersection is S
= 0, point of intersection is S, an end point of one of the lines
= -1, no point of intersection

EPS(2) - Calculation significance tolerance (real, double precision): EPS(1) for area, angle calculations; EPS(2) for length calculations.

$$X \equiv 0 \quad \text{if} \quad |X| \leq \varepsilon$$

Method: $K \geq 0$ returned only when point of intersection, S, exists; e.g., non-parallel line segments do intersect at S or segments of same parallel line have some segment in common (or are within ε_2 of each other). K = 0 is returned when distance between S and any end point of AAA or BBB is $< \varepsilon_2$.

Design Requirements: All lines are defined in two dimensions only; i.e., only (x,y) values are considered; all real variables supplied to and returned from PTINTR, as well as all internal calculations, are double precision.

Subroutine LØCPT (LOCate PoinT)

Entry Point: None

Purpose: Determines location of each of N points, P, relative to surface bounded by M points, S.

Calling Sequence: CALL LØCPT(N,P,M,S,K,KS,EPS,LØC)

N - Number (≤ 4) of points to locate (integer)

P(3,4) - Coordinates of points to locate (real, double precision)

m - Number of points (≤ 4) describing boundary of surface.

S(3,4) - Coordinates of surface bounding points (real, double precision)

K(2,4) - Indices (within S) of end point of surface edges

KS(3) - Unit vector normal to surface (real, double precision)

EPS(2) - Calculation significance tolerance (real, double precision): EPS(1) for area, angle calculation; EPS(2) for length calculations.

$$X \equiv 0 \quad \text{if} \quad |X| \leq \varepsilon$$

LØC(4) - Location indication for each point tested (integer):

= 1, point inside surface boundary

= 0, point on surface boundary

= -1, point outside surface boundary

Method: VP, the vector from leading end point of each surface edge to point to locate is tested against the vector along surface edge.

The point is within surface boundary (LØC = 1) when it is not 'outside' any edge or 'on' any edge.

The point is on a surface edge (LØC = 0) when:

$$|VP| \leq \varepsilon_2$$

The point is outside a surface edge (LØC = -1) when:

$$(VE \times VP) \cdot k < -\varepsilon_1 \quad \text{or}$$

$$(VE \cdot VP) \leq \varepsilon_2 \quad \text{or}$$

$$|VE| + \varepsilon_2 < |VP|$$

Design Requirements/Limits: All real arguments and computations are double precision. Not more than four points may be positioned relative to a surface with not more than four edges (bounded by not more than four points).

Subroutine PØLYPT (PØLYgon PoinTs)

Entry Points: None

Purpose: Determine coordinates of points describing the polygon of area common to a triangle (structural element) and another figure (fluid element) of three or four sides.

Calling Sequence: CALL PØLYPT(LØCTØF,STEDGE,TR,NGRIDF,FLEDGE,FL,LØCFØS, EPS,NPØLY,P)

LØCTØF(3) - Indication of position of triangle point relative to other figure (see Subroutine LØCPT) (integer).

STEDGE(2,3) - Indices of end points of triangle edges (in array TR) (Integer).

TR(3,3) - Rectangular coordinates of structural element (real, double precision).

NGRIDF - Number of points for fluid element (integer).

FLEDGE(2,4) - Indices (in array FL) of endpoints of fluid element edges (integer).

FL(3,4) - Rectangular coordinates of fluid element points (real, double precision).

LØCFØS(4) - Indication of position of fluid points relative to structural element (integer).

EPS(2) - Calculation significance tolerance (real, double precision): EPS(1) for area, angle calculation; EPS(2) for length calculations.

$$X \equiv 0 \quad \text{if} \quad |X| \leq \epsilon$$

NPØLY - Number of points (≤ 7) found to describe polygon of common area (integer).

P(2,7) - x,y coordinates of points describing polygon of common area (real, double precision).

ethod: Starting with first triangle edge, successive points of intersection of (structural) triangle edges and fluid element edges are obtained (by Subroutine PTINTR).

Design Requirements/Limitations: All real arguments and computations are double precision.

Function DAPØLY (Double Precision Area of a PØLYgon)

Entry Points: None

Purpose: Obtain area of a polygon.

Calling Sequence: A = DAPØLY(N,P)

N ≤ 10 - Number of points describing polygon (integer, single precision).

P(2,N) - x,y coordinates of points describing polygon (real, double precision).

Method:

$$A = -\oint y\, dx$$

Contribution from side whose endpoints are P_i, P_j is:

$$A_{ij} = 1/2\ [(y_i + y_j) * (x_i - x_j)]$$

Design Considerations:

1. Polygon must not have more than 10 sides.
2. All real arguments and internal computations are double precision.

Function DVMAG (Double Precision Vector MAGnitude)

Entry Points: None

Purpose: Obtains magnitude of a vector.

Calling Sequence: X = DVMAG(V,EPS)

V(3) - i,j,k coefficients of vector, Vl (real, double precision)

EPS - Calculation significance tolerance (real, double precision): EPS(1) for area, angle calculation; EPS(2) for length calculations.

$$X \equiv 0 \quad \text{if} \quad |X| \leq \varepsilon$$

Method:

$$A = V \cdot V$$

$$DVMAG = \sqrt{A} \quad \text{if } A > \varepsilon > 0$$

$$= 0 \quad \text{if } A \leq \varepsilon$$

Design Requirements: All real arguments and internal computations are double precision.

Subroutine DCRØSS (Double precision CRØSS product)

Entry Point: DNØRM

Purpose: Obtain cross-product of two vectors.

Calling Sequence: CALL DCRØSS(X,Y,Z)

X(3),Y(3),Z(3) - i,j,k coefficients of vectors $\bar{X},\bar{Y},\bar{Z}$.

Method: $\bar{Z} = \bar{X} \times \bar{Y}$

Design Considerations: All arguments and internal computations are double precision.

Subroutine DNØRM (Double precision NØRMalization of a vector quantity)

Entry Point: DCRØSS

Purpose: Normalize a vector to its magnitude.

Calling Sequence: CALL DNØRM(X,MAG)

X(3) - i,j,k coefficients of vector $\bar{X}$ (real, double precision).
MAG - Magnitude of vector $\bar{X}$ (real, double precision).

Method:

$$A = X_i^2 + X_j^2 + X_k^2$$

$$MAG = A, \quad A > 0$$

$$\bar{X} = \frac{X_i}{MAG}, \frac{X_j}{MAG}, \frac{X_k}{MAG}$$

Design Considerations:

1. DNØRM returns un-normalized vector quantity, $\bar{X}$, when $A \leq 0$.
2. All arguments and internal computations are double precision.

Diagnostic Message

Messages 8000 through 8014 may be issued by this module.

4.3 FUNCTIONAL MODULE GFSMA (GENERAL FLUID/STRUCTURE MATRIX ASSEMBLER)

Entry Point: GFSMA

Purpose

To assemble the stiffness, mass and pressure transformation matrices in a hydroelastic analysis.

DMAP Calling Sequence

```
GFSMA   AXY,AFRY,KYY,DKAA,DKRFFR,KAA,MAA,GM,GØ,USET,USETF,PHIA,PHIX,
        LAMA/KMAT,MMAT,GIA,PØUT,HC/V,N,NØGRAV/V,N,NØFREE/V,Y,KCØMP/
        V,Y,CØMPTYP/V,N,FØRM/V,Y,LMØDES $
```

Input Data Blocks

AXY	- Structure/fluid area matrix	
AFRY	- Free surface area matrix	
KYY	- Fluid stiffness matrix	
DKAA	- Structure gravity stiffness matrix	
DKFRFR	- Free surface gravity stiffness matrix	
KAA	- Reduced structure stiffness matrix	Direct only
MAA	- Reduced structure mass matrix	Direct only
GM	- Multipoint constraint transformation matrix	Direct only
GØ	- Omit point transformation matrix	Direct only
USET	- Structure only set definition table	Direct only
USETF	- Fluid and structure set definition table	
PHIA	- Solution eigenvectors, A-set	Modal only
PHIX	- Solution eigenvectors, X-set	Modal only
LAMA	- Solution eigenvalue table	Modal only

Notes

1. AFRY and DKFRFR may be purged if no free surface points exist.
2. DKFRFR and DKAA may be purged if no gravity exists.
3. GM may be purged if no multipoint constraints exist.
4. GØ may be purged if no omit points exist.
5. PHIA, PHIX, and LAMA may be purged if the direct formulation is used.
6. KAA, MAA, GM, GØ, and USET may be purged if the modal formulation is used.

Output Data Blocks

KMAT - Combination fluid/structure stiffness matrix

MMAT - Combination fluid/structure mass matrix

GIA - Pressure transformation matrix

PØUT - Partitioning vector for the modal displacements

HC - Constraint transformation matrix for incompressible calculations.

Notes

1. PØUT will be purged if the direct formulation is used.
2. HC will be purged if the incompressible calculations are not used.

Parameters

NØGRAV - Input - integer - no default. Flag which specifies whether gravity effects are present (-1 no gravity).

NØFREE - Input - integer - no default. Flag which specifies whether free surface exists (-1 implies no free surface).

KCØMP - Input - real - default = 1.0. Compressibility factor.

CØMPTYP - Input - integer - default = -1. Type of compressibility calculation to be used:

-1 = structure and free surface are coupled with a spring to resist volume change

+1 = incompressible - constraint equation is generated to restrict volume change

FØRM - Input - integer - default = 01. Type of formulation to be used:

-1 = direct formulation

+1 = modal formulation

LMØDES - Input - integer - default = -1. Number of structure modes to be used in the modal formulation (-1 indicates all available modes are used).

Method

Subroutine GFSMA is a small driver which calls the following routines based on the input parameters:

1. GFSDIR - assembles the combined stiffness and mass matrices using the direct formulation.

2. GFSMØD - assembles the combined stiffness and mass matrices using the modal formulation.

Subroutines

Subroutine Name: GFSDIR

Entry Point: GFSDIR

Purpose: To assemble the fluid matrices using the direct formulation method.

Calling Sequence: CALL GFSDIR

Method:

1. The structure/fluid area matrix is reduced to a set using the following steps.

   If MPC's are present, the MPC points are partitioned out.

$$[A_{xy}] \Rightarrow \begin{bmatrix} \bar{A}_{ny} \\ \hline A_{my} \end{bmatrix}$$

$$[A_{ny}] = [\bar{A}_{ny}] + [G_m]^T [A_{my}]$$

   If SPC's are present, the SPC points are partitioned out.

$$[A_{ny}] \Rightarrow \begin{bmatrix} \bar{A}_{fy} \\ \hline A_{sy} \end{bmatrix}$$

   If omits are present:

$$[A_{fy}] \Rightarrow \begin{bmatrix} A_{ay} \\ \hline A_{oy} \end{bmatrix}$$

$$[A_{ay}] = [\bar{A}_{ay}] + [G_o]^T [A_{oy}]$$

   Subroutines GFSPTN, SSG2B, and CALCV are utilized to perform these calculations.

2. If free surface points exist, they are merged with the reduced area matrix using subroutine GFSMRG.

$$\begin{bmatrix} A_{ay} \\ \hline A_{fry} \end{bmatrix} \Rightarrow [A_{wy}]$$

3. If SPC points exist on the fluid they are partitioned out from the area and fluid stiffness matrices

$$[A_{wy}] \Rightarrow [A_{wj} \mid A_{ws}]$$

and this matrix is transposed using subroutine GFSTRN.

$$[A_{jw}] = [A_{wj}]^T$$

$$[K_{yy}] \Rightarrow \left[\begin{array}{c|c} K_{jj} & K_{js} \\ \hline K_{sj} & K_{ss} \end{array}\right]$$

4. If no SPC points exist on the fluid, the first fluid point is reduced out to remove potential singularities.

$$[K_{yy}] \Rightarrow \left[\begin{array}{c|c} K_{11} & K_{1j} \\ \hline K_{j1} & K_{jj} \end{array}\right]$$

generate [H] ($N_{(y-1)}$ by N_y transformation) using subroutine GFSH.

$$[H] = \frac{1}{N}\begin{bmatrix} -1 & (n-1) & -1 & -1 & \cdots \\ -1 & -1 & (n-1) & -1 & \cdots \\ & & \vdots & & \end{bmatrix}$$

$$[A_{jw}] = [H][A_{wy}]^T$$

For CØMPTYP < 0, the compressibility factor is then generated using subroutine GFSCØM. This matrix contains the spring used to restrict volume change.

$$\{A_c\} = [A_{wy}]\{I\}$$

$$[K_c] = KCØMP \{A_c\} \lfloor A_c \rfloor$$

For CØMPTYP > 0 a constraint equation is generated to restrict volume change. Subroutine GFSHC performs these calculations.

$$\{A_c\} = [H]\{I\}$$

The largest row in A_c,m, is then chosen. The m^{th} row of matrix H_c is null and all other columns, i, have a 1.0 on the diagonal and $-A_{ci}/A_{cm}$ in row m.

$$[H_c] = \left[\begin{array}{c|c} 0 & -A_{ci}/A_{cm} \\ \hline 0 & I \end{array}\right]$$

5. The pressure transformation matrix is then found by solving the following equation:

$$[K_{jj}][G_{jw}] = [A_{jw}]$$

Matrix utilities FACTØR and SSG3A are used to perform this solution.

6. If gravity exists, the additional stiffness is added in

$$[\bar{K}_{aa}] = [K_{aa}] + [DK_{aa}]$$

otherwise,

$$[\bar{K}_{aa}] = [K_{aa}]$$

7. The free surface stiffness is then merged in with subroutine GFSMRG if it exists.

$$\left[\begin{array}{c|c} \bar{K}_{aa} & 0 \\ \hline 0 & DK_{frfr} \end{array}\right] \Rightarrow [\bar{K}_{ww}]$$

$$\left[\begin{array}{c|c} M_{aa} & 0 \\ \hline 0 & 0 \end{array}\right] \Rightarrow [\bar{M}_{ww}]$$

8. The combined stiffness and mass matrices are computed.

$$[M_{ww}] = [\bar{M}_{ww}] + [A_{jw}]^T[G_{jw}]$$

If no SPC's existed on the fluid for CØMPTYP < 0

$$[K_{ww}] = [\bar{K}_{ww}] + [K_c]$$

for CØMPTYP > 0

$$[K_{ww}] = [H_c]^T[\bar{K}_{ww}][H_c]$$

$$[MT] = [H_c]^T[M_{ww}][H_c]$$

a 1.0 is then added to row m column m of [MT] using subroutine GFSMT.

$$[M_{ww}] = [MT] + \left[\begin{array}{c|c} 1.0 & 0 \\ \hline 0 & 0 \end{array}\right]$$

otherwise,

$$[K_{ww}] = [\bar{K}_{ww}], \quad [M_{ww}] = [MT]$$

and $[K_{ww}]$ is written on data block KMAT and $[M_{ww}]$ is written on data block MMAT.

9. The final pressure transformation matrix is created by expanding the j size matrix to y size and then removing free surface points.

If no SPC's exist on the fluid,

$$[G_{yw}] = [H]^T[G_{jw}]$$

otherwise merge zeros in,

$$\left[\begin{array}{c} G_{jw} \\ \hline 0 \end{array}\right] \Rightarrow [G_{yw}]$$

finally partition out the free surface

$$[G_{yw}] \Rightarrow \begin{bmatrix} G_{frw} \\ \hline G_{iw} \end{bmatrix}$$

$[G_{iw}]$ is then written on data block GIA.

Subroutine Name: GFSMØD

Entry Point: GFSMØD

Purpose: To assemble the fluid matrices using the modal formulation method.

Calling Sequence: CALL GFSMØD

Method:

1. A dummy uset vector, USETD, is created which contains the following bit positions:

   U_m - modal point, UA + UNZ
   U_ξ - desired modal point (based on LMØDES parameter)
   $U_{n\xi}$ - modal point to be skipped
   U_{fr} - free surface point
   U_h - $U_{fr} + U_\xi$

   Files PHIX and USETF and parameter LMØDES are used to set these bits.

2. The desired modes are partitioned from the PHIX matrix using the USETD vector.

$$[PHIX] \Rightarrow [\phi_{x\xi} \mid \phi_{xn\xi}]$$

3. The area matrix is then transformed.

$$[A_{\xi y}] = [\phi_{x\xi}]^T [A_{xy}]$$

4. If the free surface points exist, they are merged in using subroutine GFSMRG.

$$\left[\begin{array}{c} A_{\xi y} \\ \hline A_{fry} \end{array}\right] \Rightarrow [A_{hy}]$$

5. If SPC points exist on the fluid they are partitioned out from the area and fluid stiffness matrix.

$$[A_{hy}] \Rightarrow [A_{hj} \mid A_{ns}]$$

and this matrix is transposed using subroutine GFSTRN.

$$[A_{jn}] = [A_{hi}]^T$$

$$[K_{yy}] \Rightarrow \left[\begin{array}{c|c} K_{ii} & K_{js} \\ \hline K_{si} & K_{ss} \end{array}\right]$$

6. If no SPC points exist on the fluid, the first fluid point is reduced out to remove potential singularities.

$$[K_{yy}] = \left[\begin{array}{c|c} K_{11} & K_{1i} \\ \hline K_{i1} & K_{ii} \end{array}\right]$$

The [H] matrix is then generated using subroutine GFSM. ($N_{(y-1)}$ by N_y transformation)

$$[H] = \frac{1}{N}\begin{bmatrix} -1 & (n-1) & -1 & -1 & \cdots \\ -1 & -1 & (n-1) & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

$$[A_{jn}] = [H][A_{hy}]^T$$

The compressibility matrix which contains the spring is generated to restrict volume change.

$$\{A_c\} = [A_{hy}]\{I\}$$

$$[K_c] = KC\emptyset MP\ \{A_c\}[A_c]$$

7. The pressure transform matrix is then found by solving the following equation:

$$[K_{ji}][G_{jh}] = [A_{jh}]$$

Matrix utilities FACTØR and SSG3A are used to perform this solution.

8. The generalized stiffness and mass terms are then extracted from the LAMA data block to form the diagonal modal mass and stiffness matrices.

$$[K_{\xi\xi}] = \begin{bmatrix} K_1 & & \\ & \ddots & \\ & & K_N \end{bmatrix}$$

$$[M_{\xi\xi}] = \begin{bmatrix} M_1 & & \\ & \ddots & \\ & & M_n \end{bmatrix}$$

9. If free surface points exist, the modal mass matrix is expanded and the final mass matrix is formed.

$$\left[\begin{array}{c|c} M_{\xi\xi} & 0 \\ \hline 0 & 0 \end{array}\right] \Rightarrow [\bar{M}_{hh}] \qquad (h = \text{modes + free surface})$$

$$[M_{hh}] = [\bar{M}_{hh}] + [A_{jh}]^T[G_{jh}]$$

and $[M_{hh}]$ is written on data block MMAT.

10. If gravtiy exists, the additional stiffness is transformed and added in.

$$[PHIA] \Rightarrow [\phi_{a\xi} \quad \phi_{an\xi}]$$

$$[\bar{K}_{\xi\xi}] = [K_{\xi\xi}] + [\phi_{a\xi}]^T[DK_{aa}][\phi_{a\xi}]$$

otherwise

$$[\bar{K}_{\xi\xi}] = [K_{\xi\xi}]$$

11. The free surface stiffness is then merged in if it exists.

$$\left[\begin{array}{c|c} \bar{K}_{\xi\xi} & \\ \hline & DK_{frfr} \end{array}\right] \Rightarrow [\bar{K}_{hh}]$$

12. The final stiffness matrix is then computed by adding in the compressibility term if it exists.

If no SPC's existed on the fluid

$$[K_{hh}] = [\bar{K}_{hn}] + [K_c]$$

otherwise

$$[K_{hh}] = [\bar{K}_{hh}]$$

and $[K_{hh}]$ is written on data block KMAT.

13. The final pressure transformation matrix is created by expanding the j size to h and then removing the free surface points.

If no SPC's exist on the fluid,

$$[G_{yh}] = [H]^T[G_{jh}]$$

otherwise zeros are merged in.

$$\left[\begin{array}{c} G_{jh} \\ \hline 0 \end{array}\right] \Rightarrow [G_{yh}]$$

Finally the free surface is removed.

$$[G_{yh}] \Rightarrow \left[\begin{array}{c} G_{frh} \\ \hline G_{ih} \end{array}\right]$$

$[G_{ih}]$ is written on data block GIA.

Subroutine Name: GFSSPC

Entry Point: CFSSPC (in GFSUTL)

Purpose: To calculate a partitioning vector to remove the first row and columns of the fluid stiffness matrix if no SPC's are provided.

Calling Sequence: CALL GFSSPC(NUY,PVEC)

NUY - Size of fluid stiffness matrix
PVEC - Gino file number of partitioning vector

Subroutine Name: GFSCØM

Entry Point: GFSCØM (in GFSUTL)

Purpose: To compute the fluid compressibility matrix for the case where no SPC's exist on the fluid. This matrix contains the spring factors that couple the free surface and the structure points to resist volume changes.

Calling Sequence: CALL GFSCØM(AWY,NUY,KC,IDENT,AC,SCR)

AWY - Gino file number of the AWY matrix, direct formulation, or AMY matrix, modal formulation
NUY - Size of fluid matrices
KC - Gino file number of the compressibility matrix to be calculated
IDENT - Gino file number of a scratch file
AC - Gino file number of a scratch file
SCR - Gino file number of a scratch file

Method: See Step 4 under subroutine GFSDIR for the calculations used to generate KC.

Subroutine Name: GFSH

Entry Point: GFSH (in GFSUTL)

Purpose: To calculate the H transformation matrix used when the first row and column of the fluid stiffness matrix is removed.

Calling Sequence: CALL GFSH(NUY,H)

NUY - Size of fluid stiffness matrix

H - Gino file number for H matrix

Method: See Step 4 under subroutine GFSDIR for the calculations used to generate H.

Subroutine Name: GFSHC

Entry Point: GFSHC (in GFSUTL)

Purpose: To generate the constraint transformation matrix for the pure incompressible calculations when requested by the user. These constraints will restrict any volume changes by the fluid.

Calling Sequence: CALL GFSHC(AWY,NUY,HC,IDENT,AC,MRØW)

AWY - Gino file number of the AWY matrix

NUY - Size of fluid stiffness matrix

HC - Gino file number of the HC matrix

IDENT - Gino file number of a scratch matrix

AC - Gino file number of a scratch matrix

MRØW - Row number of the largest term in the calculated AC matrix

Method: See Step 4 under subroutine GFSDIR for the calculations used to generate HC.

Subroutine Name: GFSMT

Entry Point: GFSMT (in GFSUTL)

Purpose: To add a 1.0 value to the null row and column of the fluid structure mass matrix to prevent singularities. The routine is only used when the incompressible calculations are requested.

Calling Sequence: CALL GFSMT(MT,MMAT,MRØW)

MT - Gino file number of the input mass matrix

MMAT - Gino file number of the final mass matrix

MRØW - Null row and column number. Output from subroutine GFSHC.

Method: Subroutine CPYSTR is used to copy each column up to column MRØW from matrix MT to MMAT. A 1.0 is then packed in matrix MMAT in row position MRØW using subroutine ZBLPKI. The remainder of matrix MT is then copied to MMAT using subroutine CPYFIL.

Subroutine Name: GFSPTN

Entry Point: GFSPTN

Purpose: General purpose partition routine to perform the following partition:

$$[A] \Rightarrow \left[\begin{array}{c|c} A11 & A12 \\ \hline A21 & A22 \end{array}\right]$$

Calling Sequence: CALL GFSPTN(A,A11,A21,A12,A22,RPART,CPARY)

A - Gino file number of matrix to be partitioned

A11, A21, A12, A22 - Gino file numbers of output matrices. Any of these may be zero if that partition is not desired.

RPART - A partitioning vector whose length is the size of a row in A. It is used to partition the columns of A and may be zero if A12 and A22 do not exist.

CPART - A partitioning vector whose length is the size of a column in A. It is used to partition the rows of A and may be zero if A21 and A22 do not exist.

Method: The input arguments are used to initialize common block /PARMEG/ and subroutine PARTN is called to perform the actual partition operation.

Subroutine Name: GFSMRG

Entry Point: GFSMRG

Purpose: General purpose merge routine to perform the following merge:

$$\left[\begin{array}{c|c} A11 & A12 \\ \hline A21 & A22 \end{array}\right] \Rightarrow [A]$$

Calling Sequence: CALL GFSMRG(A,A11,A21,A12,A22,RPART,CPART)

Argument values are the same as those discussed under subroutine GFSPTN.

Method: The input parameters are used to initialize common block /PARMEG/ and subroutine MERGE is called to perform the actual merge operation.

Subroutine Name: GFSTRN

Entry Point: GFSTRN

Purpose: To transpose a matrix.

Calling Sequence: CALL GFSTRN(A,AT,I,SCR)

A - Gino file number of matrix to be transposed
AT - Gino file number of the transpose
I - Gino file number of a scratch file
SCR - Gino file number of a scratch file

Method: Subroutine SSG2B is used to find the transpose by solving the following equation:

$$[AT] = [A]^T[I]$$

where I is an identity matrix.

This procedure is more efficient at finding the transpose then subroutine TRANP1 when matrix A is sparse.

Subroutine Name: GFSWCH

Entry Point: GFSWCH

Purpose: To switch the units on which two files reside.

Calling Sequence: CALL GFSWCH(FILE1,FILE2)

FILE1, FILE2 } Gino file numbers of files to be switched.

Method: The switch is performed by updating the FIST and FIAT to point to the new files. A check is made to ensure that any stacked data blocks in the FIAT are also changed to reflect the switch.

Diagnostic Messages

A number of messages produced by the matrix utilities can be issued by this module.

4.4 UTILITY MODULE 'TRAILER'

Name: TRAILER

Purpose

To examine or modify the trailer of a GINØ data block.

DMAP Calling Sequence

```
TRAILER   A//C,N,opt/C,N,word/5,N,value $
```

Input Data Blocks

A - Data block for which the trailer is desired

Output Data Blocks: None

Parameters

1. opt is a BCD operation code from the list below. (input, no default)

   RETURN - The value of the specified trailer word is to be returned
   STØRE - The value of the specified trailer word is to be changed

2. word is an integer value which specifies the particular trailer word to be operated on. The value of this parameter must be $1 \leq word \geq 6$. (input, no default)

3. value is an integer parameter which will contain the value of specified trailer word, opt = RETURN, or contains the value to be stored in the trailer, opt = STØRE. If specified, the range must be $0 \leq value \leq 65535$. (input, no default)

Remarks

1. For matrix data blocks, the trailer positions contain:

   WORD 1 - Number of columns
   WORD 2 - Number of rows
   WORD 3 - Matrix form
   WORD 4 - Type of matrix elements
   WORD 5 - Maximum number of non-zero words in any one column
   WORD 6 - Matrix density x 10^4

2. If the data block is purged, the parameter value will be returned negative.

Examples

```
TRAILER   M1//C,Y,RETURN/C,Y,1/V,Y,CØL $
SAVE      CØL $
TRAILER   M2//C,Y,STØRE/C,Y,3/C,Y,1 $
```

4.5 DMAP EXECUTIVE OPERATION MODULE CØMPØN

Name: CØNDØN (compilation on)

Purpose

To allow blocks of DMAP statements to be compiled or skipped depending on bulk data parameter value.

DMAP Calling Sequence

```
CØMPØN   n,param $
CØMPØN   c,param $
```

where:

1. n is a BCD name of a label which specifies the end of the DMAP statement block.
2. c is an integer constant which specifies the number of DMAP statements in the block.
3. param is the name of parameter that appears on a PARAM bulk data card.

Example

```
CØMPØN   END,P1 $
MØDULE1  A/B/V,Y,PN $
   .
   .
   .
MØDULEN  A/B/V,Y,PN $
LABEL    END $

CØMPØN   2,P1 $
MØDULE1  A/B/V,Y,PN $
MØDULE2  A/B/V,Y,PN $
```

Remarks

1. The block of DMAP statements specified by the label or count is skipped if the value of parameter is false (param $\geq$ 0). If the parameter value is true (param < 0), the block of DMAP statements will be compiled.
2. If no PARAM bulk data card is provided, a value of false is assumed.

3. If the form of CØMPØN specifying a label is used, the label may not be specified by any other DMAP instructions including other CØMPØN or CØMPØFF instructions.

4. Comment cards are not included in the statement count.

5. CØMPØN and CØMPØFF instructions may be nested up to five levels using the same rules as for a FØRTRAN DØ loop.

4.6 DMAP EXECUTIVE OPERATION MODULE CØMPØFF

Name: CØMPØFF (compilation off)

Purpose

To allow blocks of DMAP statements to be compiled or skipped depending on a bulk data parameter value.

DMAP Calling Sequence

```
CØMPØFF  n,param $
CØMPØFF  c,param $
```

where:

1. n is a BCD name of a label which specifies the end of the DMAP statement block.
2. c is an integer constant which specifies the number of DMAP statements in the block.
3. param is the name of parameter that appears on a PARAM bulk data card.

Example

```
CØMPØFF   END,P1 $
MØDULE1   A/B/V,Y,PN $
   :
MØDULEN   A/B/V,Y,PN $
LABEL     END $
CØMPØFF   2,P1 $
MØDULE1   A/B/V,Y,PN $
MØDULE2   A/B/V,Y,PN $
```

Remarks

1. The block of DMAP statements specified by the label or count is skipped if the value of the parameter is true (param < 0). If the parameter value is false (param ≥ 0), the block of DMAP statements will be compiled.

2. If no PARAM bulk data card is provided, a value of false is assumed.

3. If the form of CØMPØFF specifying a label is used, the label may not be specified by any other DMAP instructions including other CØMPØN or CØMPØFF instructions.

4. Comment cards are not included in the statement count.

5. CØMPØN and CØMPØFF instructions may be nested up to five levels using the same rules as for a FØRTRAN DØ loop.

4.7 MODIFICATIONS TO EXISTING NASTRAN SUBROUTINES

Changes to the standard Level 16 were required to perform the following operations:

1. The three-dimensional fluid elements are connected to special fluid grid points with one degree of freedom. Implementing these elements required changes to the IFP, GP1, TA1, PLØT, and EMG modules. The fluid material definition required changes to the PREMAT utility subroutine.

2. Generating plots of the free surface displacements required changes to the SDR2 module to convert scalar displacements to vectors normal to the free surface.

3. The two-step eigenvalue processing in the modal formulation required changes to the CASE module to process separate subcases for eigenvalue methods.

4. The 'GIVENS' option for eigenvalue extraction in the READ module was modified to provide more efficient processing and to correct for existing errors. Specifically, the double precision operations were corrected to provide consistent calculations and the 'spill logic' in the tridiagonalization procedure was completely recoded to eliminate unnecessary core transfers and I/O operations.

4.56 FUNCTIONAL MODULE CASE (SIMPLIFY CASE CONTROL)

4.56.1 Entry Point: CASE

4.56.2 Purpose

To remove looping considerations from later dynamics modules.

4.56.3 DMAP Calling Sequence

```
CASE    CASECC,PSDL/CASEXX/C,N,APPRØACH/V,N,REPEAT/V,N,LØØP  $
```

4.56.4 Input Data Blocks

CASECC - Case Control Data Table.

PSDL - Power Spectral Density List.

Note: PSDL is used only if APPRØACH = FREQRESP and Random Analysis is selected in CASECC.

4.56.5 Output Data Blocks

CASEXX - Case Control data table for dynamics problems.

Note: CASEXX cannot be purg

4.56.6 Parameters

APPRØACH - Input-BCD-no default. Defines the approach to be used for looping criteria.

BCD Value	LØØP
STATICS	NØNE
REIGEN	~~NØNE~~ EIGENVALUE EXTRACTION METHOD
DSO	NØNE
DS1	NØNE
FREQRESP	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS
TRANRESP	LØADS
BLKO	NØNE
BLK1	NØNE
CEIGEN	DIRECT INPUT MATRICES OR TRANSFER FUNCTIONS

BCD Value	LØØP
PLA	NØNE

REPEAT - Input and output-integer-set equal to zero outside of the DMAP loop by the PARAM module. -1 if no additional loops; + loop count if loops.

LØØP - Output-integer-default = -1. -1 if this is not a looping problem, 0 if this is a looping problem.

4.56.7 Method

The method of operation depends upon the input parameter APPRØACH.

4.56.7.1 Transient Response

If APPRØACH = TRANRESP, CASECC is skipped over REPEAT records. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. An attempt is made to read another record. If no more records exist, REPEAT is set to -1. Also, if this is the first entry to CASE (i.e., REPEAT = 1), LØØP is set to -1. If additional records exist, REPEAT and LØØP are set to 1.

4.56.7.2 Complex Eigenvalue Analysis

If APPRØACH = CEIGEN, REPEAT records are skipped in CASECC. If REPEAT = 0, REPEAT is set to 1. One record of CASECC is read and copied onto CASEXX. The names of the Direct Input Matrices and Transfer Functions sets are saved. An attempt is made to read another record. If no more exist, REPEAT is set to -1. Also if this is the first entry (i.e., REPEAT = 1) LØØP is set to -1. If additional records exist, their Direct Input Matrices and Transfer Functions sets are compared to those saved. If they all agree, this record is copied onto CASEXX and the process is repeated. If they do not agree, REPEAT is incremented by 1, LØØP is set to 1, and CASE returns.

4.56.7.3 Frequency Response

If APPRØACH = FREQRESP, the method used is the same as Complex Eigenvalue Analysis except a test is also made for frequency set selection changes. In addition, if RANDPS cards are selected, the selected set is read from PSDL and the unique subcase "id's" referenced are stored. Each subcase id copied onto CASEXX is compared to this list, and the entry is marked as found. If at the completion of CASE unmarked entries exist, the routine terminates with message 3033.

4.56.7.4 Real Eigenvalues

If APPRØACH = REIGEN, the same method is used except the METHOD ID is checked.

The "stress" data recovery for heat transfer analysis is performed by subroutines SDHTF1, SDHTFF, and SDHTF2 of module SDR2. In Phase 1, the matrix K and the matrix C are calculated where K is the 3 x 3 material matrix and C_e is a 3 by number of points matrix. For the tetrahedron:

$$[C_e] = \begin{bmatrix} H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix}$$

For the WEDGE, HEXA1, and HEXA2 elements, the [C] matrix is calculated for each subelement. The column corresponding to each point of the tetrahedron is added to the column of the C_e matrix corresponding to that point in the whole element. The results are divided by the number of subelements.

In Phase 2, the temperature gradient vector and flux vector are calculated with the equations:

$$\{\Delta T\} = [C]\{u\}$$

$$\{q\} = -[K]\{\Delta T\},$$

where {u} is the vector of temperatures of the connected points.

8.17.3 Hydroelastic Calculations for Solid Elements

The "stiffness" matrix for hydroelastic problems is generated for the FHEX1, FHEX2, FTETRA, and FWEDGE elements (which are identical in format to the solid structure equivalents). The matrix is identical to the heat transfer matrix except

$$K_{xx} = K_{yy} = K_{zz} = 1/\rho$$

$$K_{xy} = K_{xz} = K_{yz} = 0$$

The density, ρ, is obtained from the MAT subroutine with INFLAG = 9.

INFLAG = 8 -- INFLAG = 8 is used only by two-dimensional element subroutines in modules PLA3 and PLA4. The fourth word of /MAIN/, PLAARG (see below), is stress (σ) and is used as the ordinate in an inverse interpolation table look-up to obtain the abscissa which is strain (ε).

If either: a) the ordinate is in the range of the piecewise linear function defined by the table on a TABLES1 bulk data card, or b) the ordinate is greater than the maximum (which is also the last) ordinate in the table but the slope of the line segment joining the last two points of the table is nonzero, then the second word of /MATØUT/ is set to zero and the abscissa, obtained by inverse linear interpolation or extrapolation, is stored in the first word of /MATØUT/. If either: a) the ordinate is less than the minimum (which is also the first) ordinate in the table, or b) the ordinate is greater than the maximum ordinate in the table and the slope of the line segment joining the last two points of the table is zero, then the integer "1" is stored in the second word of /MATØUT/ (and the first word of /MATØUT/ is set to zero). Only MAT1 cards are searched to match the input MATID.

TEMP - Average element temperature. Used as the independent variable in a table look-up when it is determined that a material property is temperature dependent. Not used when INFLAG = 5 or 6.

PLAARG - Element strain. Used as the independent variable in a table look-up when E, the modulus of elasticity, is defined as the first derivative of a strain-stress curve. Used only in the Piecewise Linear Analysis Rigid Format and only by modules PLA3 and PLA4.

SINTH - Sine of the material property orientation angle. Used only when INFLAG = 2 and the MATID is found among the MAT2 cards. Used to construct the [U] matrix referenced above.

CØSTH - Cosine of the material property orientation angle. The comments on SINTH, above, also apply here.

CØMMØN/MATØUT/ - (output Common Block). Length 20 words. Depending upon the values of INFLAG, the output common block is defined variously as follows:

INFLAG = 9 -- Is used by hydroelastic 3-D fluid elements. The value of density (ρ) is returned in /MATØUT/.

8. Strain Functional Value (INFLAG = 8)

Word	Symbol	Definition
1	PLAANS	Value of strain (ε) as an inverse function of stress (σ)
2	ICELL2	= 0 if the input stress is in the range of the function = 1 if the input stress is outside the range of the function
3-26		Undefined

3.4.30.4 Method

1. PREMAT: All the MAT1, MAT2, ~~and~~ MAT3 and MATF cards are read from the MPT data block into open core so that each card is assigned 1 + 3*N words of core where N, a function of the card type, is the number of material property data items on that card type. The first word is the material identification number and each material property is allocated 3 words: the first the input material property; the second a table (function) number which gives this material property as a function of temperature; the third a table number which gives this material property as a function of stress. Initially words 2 and 3 are set to zero. Although the third word is currently used only for MAT1 cards and for E, the modulus of elasticity, on that card, future development may make use of a more general application of stress dependent material properties. If there are n temperature dependent material properties for a non-Piecewise Linear Analysis problem, PREMAT is wrapped up and a RETURN to the calling routine is executed.

For a non-Piecewise Linear Analysis problem for which a temperature set for material properties was selected in the user's Case Control Deck, all MATT1, MATT2 and MATT3 cards are read into open core from the MPT data block. For a Piecewise Linear Analysis problem MATS1 cards are read into open core from the MPT. A sorted list, with duplicates discarded, of the table numbers referenced on these cards is constructed in open core. This table number list is constructed so that every referenced table has eleven locations allocated to it. These eleven locations are used as a dictionary for the tables. The contents are: the table number (word 1); the table type 1,2,3, or 4 (word 2); pointers to the first and last entries in the table (words 3 and 4); parameters from the TABLE card (words 5 through 11). The DIT data block is then read. For each table read, it is determined by scanning the table number list whether or not the table is required for problem solution. If it is required, the table is read into open core and the dictionary entry for

Table 1(f). Bulk Data Cards Processed by IFP.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
291	CFFREE	8	GEØM2	0	4	8	505	1	4404	75	1		-1	
292	CFLSTR	8	GEOM2	0	-4	9	-1	1	4504	76	3		-1	
293	CFHEX1	8	GEØM2	0	16	16	531	1	4604	77	4		-1	
294	CFHEX2	8	GEØM2	0	16	16	531	1	4704	78	4		-1	
295	CFTETRA	8	GEØM2	0	8	8	337	1	4804	79	4		-1	
296	CFWEDGE	8	GEØM2	0	8	8	525	1	4904	80	4		-1	
297	MATF	3	GEØM2	0	4	8	198	1	5004	81	1		-1	

$$\{j\} = \frac{\{k\} \times \{V_i\}}{|\{k\} \times \{V_i\}|}, \text{ ("Y" unit vector);} \tag{21}$$

$$\{i\} = \{j\} \times \{k\}, \text{ ("X" unit vector).} \tag{22}$$

The orientation of the axes is defined by the matrix

$$[T_{ON}] = [T_{OM}] \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix} \tag{23}$$

5. On each pass of the CØRDij data at least one new system must be converted. After each pass the referenced GRID data is checked and converted. The resulting CØRDij data will be the CSTM data block with each entry reduced from 16 to 14 words.

4.21.7.5 Construction of the BGPDT, the SIL and the Second Logical Record of the EQEXIN.

The BGPDT and the SIL data blocks are formed simultaneously. The SIL data block is simply a list of the first scalar index for each grid or scalar point. The number of scalar indices (or degrees of freedom) for each point is determined by examining the elements connected to each point. The maximum number of degrees of freedom for each element type is listed in /GPTA1/. The maximum degrees of freedom for each point is determined by reading data block GEØM2 and examining the connection information in conjunction with the degree of freedom information in /GPTA1/.

The GPDT data are read a point at a time. The basic location coordinates of the point are formed using Equation 8 through Equation 14 and these data are written on the BGPDT file. The SIL value for the next point is calculated by incr enting the last value by six (grid point) or by one (scalar point). (or connected to a fluid element: FHEX1, FHEX2, FTETRA, or FWEDGE.

A test is made on the value of the displacement coordinate system (field 6) in the GPDT data. If this value is the integer, -1, the point is a special RINGFL, GRIDF, or GRIDS fluid point. It is given one scalar index, the displacement coordinate system is basic (0), and its location coordinates in the BGPDT data block are calculated like a normal grid point.

Finally the second logical record of EQEXIN is written. This record contains pairs of external numbers, 10*scalar index + type where type = 1 for a grid point, 2 for a scalar point.

opened, if not yet opened, and the next vector present is unpacked into core.

Data items are now assembled for the identification record, and this identification record is output to the output data block. Output line entries for the point-ID's requested are then written on the output data block forming a data record. At this time, if the user requested magnitude/phase for complex outputs, the magnitude/phase computations are performed on the real/imaginary pairs.

When all requests have been processed for this vector, the next Case Control record is read. If no more Case Control records exist and there are more vectors present, those vectors are processed using the last Case Control record's specifications.

When all vectors have been processed for the current loop pass, the next pass may be made for forces of single-point constraint or loads.

If deformed structure plots are requested, an output plot data block is formed during the first loop pass, described above, containing translation components of the displacement vector rotated to basic coordinates. Scalar fluid points are identified by a value of -2 of the CID. Six degrees of freedom are generated for plotting this displacement.

4.46.8.21 Subroutine Name: SDR2D

1. Entry Point: SDR2D

2. Purpose: To perform stage V as defined above, under "Method".

3. Calling Sequence: CALL SDR2D

4. Method: SDR2D performs the phase 2 stress and force recovery computations. In this phase actual stresses and forces are computed for the user-requested elements. These stresses and forces are a function of the stress matrices computed in Stage III and the displacements at the grid points of the elements.

The operations of SDR2D are dependent upon the Rigid Format being executed. In all cases the Case Control data block is opened first. For eigenvalue problems a list of eigenvalues and mode numbers is read into core from LAMA and CLAMA. For Differential Stiffness or Buckling phase 1 problems, the first record of Case Control, which is used in phase 0 of Buckling or Differential Stiffness, is skipped. For frequency or transient response problems, a list of frequencies or times is read into core from PPF or PPT.

MØ	GINØ file number of the input matrix - integer - input.
MD, MR1, M1, M2, M3, M4	- GINØ file numbers of scratch files - integer - input.
RSTRT	- '0' indicates no restart is being made - integer - input.
N	- Order of the problem - integer - output.
LFREQ, HFREQ	- Frequency range for computation of eigenvectors - real - input.
ØRDER	- Eigenvalue sort order flag - integer - input.
LAMA, ØEIGS, PHIA	- GINØ file name of the associated data blocks - integer - input.
NV	- Number of eigenvectors to compute - integer - input.
NFØUND	- Number of rigid body modes previously found - integer - input.
NVER	- Number of fails to converge on eigenvectors - integer - output.
NEVER	- Number of fails to converge on eigenvalues - integer - output.
MAX	- Maximum number of QR iterations allowed - integer - input.
ITER	- Reason for termination - integer - input.
Z	- Open core for VALVEC.

X1, X9, X18, YY, X8 are dummy variables and are not currently used.

4.48.8.30 Subroutine Name: SMLEIG

1. Entry Point: SMLEIG

2. Purpose: To compute the eigenvalues for a 1 by 1 and 2 by 2 matrix and the eigenvector for a 1 by 1 matrix.

3. Calling Sequence: CALL SMLEIG (D,Ø,VAL)

D	- Array of diagonal values - double precision - output.
Ø	- Array of off-diagonal values - double precision - output.
VAL	- Array of eigenvalues - double precision - output.

4.48.8.31 Subroutine TRIDI

1. Entry Point: TRIDI

2. Purpose: To tridiagonalize a symmetric matrix.

3. Calling Sequence: CALL TRIDI (D,Ø,V,B,A,LØC,DA)

D - Diagonal terms of the tridiagonal matrix - double precision - output.

Ø - Off-diagonal terms of the tridiagonal matrix - double precision - output.

V - Scratch array which contains another copy of the diagonal terms at the conclusion of TRIDI - double precision - output.

A - Remainder of core - single precision - scratch.

B - Scratch array which contains the square of the off-diagonal terms - double precision - output.

DA - Remainder of core - double precision - scratch.

LØC - Remainder of core - integer - scratch.

4.48.8.32 Subroutine Name: SICØX

1. Entry Point: SICØX

2. Purpose: To initialize the arrays in SICØD and SICØS. See section 4.48.8.33.

3. Calling Sequence: CALL SICØX (SIN,CØS)

SIN - Array of sine rotation factors - double precision - input/output.

CØS - Array of cosine rotation factors - double precision - input/output.

4.48.8.33 Subroutine Name: SICØX

1. Entry Point: SICØD

2. Purpose: To compute the rotation factors for a given row in double precision.

3. Calling Sequence: CALL SICØD (RØW,D,RØT)

RØW - The number of the current row to rotate - integer - input.

RØT - If no rotations are required for this row, RØT = 0. Otherwise, RØT = 1 - integer - input.

D - Array of terms for this row - double precision - input/output.

4.48.8.34 Subroutine Name: SICØX

1. Entry Point: SICØS

2. Purpose: To compute the rotation factors for a given row in single precision.

3. Calling Sequence: CALL SICØS (RØW,DS,RØT)

RØW - The number of the current row to rotate - integer - input.

DS - Array of terms for this row - single precision - input/output.

RØT - If no rotations are required for this row, RØT = 0. Otherwise, RØT = 1 - integer.

4.48.8.35 Subroutine Name: RØTAX

1. Entry Point: RØTAX

2. Purpose: To initialize the arrays in RØTATE. See section 4.48.8.35.

3. Calling Sequence: CALL RØTAX (SIN,CØS)

SIN - Array of sine rotation factors - double precision - input.

CØS - Array of cosine rotation factors - double precision - input.

4.48.8.36 Subroutine Name: RØTAX

1. Entry Point: RØTATD

2. Purpose: To rotate as much of the matrix as fits into core using double precision arithmetic.

3. Calling Sequence: RØTATD (Ø,A,RØW,RØW1,RØW2,LØC)

Ø - Array of off-diagonal values - double precision - input/output.

A - Partition of the matrix held in core - double precision - input/output.

RØW - The row number of current rotation row - integer - input.

RØW1 - The row number of the first row of the matrix partition in core - integer - input

RØW2 - The row number of the last row of the matrix partition in core - integer - input.

LØC - For spill case a vector containing the open core location for each row currently in core - integer - input.

4.48.8.37 Subroutine Name: RØTAX

1. Entry Point: RØTATS

2. Purpose: To rotate as much of the matrix as fits into core using single precision arithmetic.

3. Calling Sequence: CALL RØTATS (ØS,AS,RØW,RØW1,RØW2,LØC)

ØS - Array of off-diagonal values - single precision - input/output.

AS - Partition of the matrix held in core - single precision - input/output.

RØW - The row number of the current row - integer - input.

RØW1 - The row number of the first row of the matrix partition in core - integer - input.

RØW2 - The row number of the last row of the matrix partition in core - integer - output.

LØC - For spill case a vector containing the open core locations for each row currently in core - integer - input.

4.48.8.38 Subroutine QRITER

1. Entry Point: QRITER

2. Purpose: To obtain the eigenvalues of a tridiagonal matrix by the Ortega - Kaiser QR iteration technique.

3. Calling Sequence: CALL QRITER (VAL,Ø,LØC,QR)

INTRODUCTION

This document is the Programmer's Manual for the MESHGEN mesh generation program. It includes descriptions of all routines used by MESHGEN as well as the data base and file operations. Additionally, a section on system considerations points out areas of code that would require change for implementation on other computer hardware.

1. MESHGEN ORGANIZATION

MESHGEN has been developed and organized in a modular manner. Such a design allows both ease in debugging and simplicity of modification. The main driver routine performs virtually no computation, but serves only as a logical framework to guide execution. The overall logic of this routine is shown in Figure 1 with annotations describing functions.

For the most part, all data is passed between major routines through common blocks and, in some cases, by scratch files. This again enhances the ability to expand and modify the system.

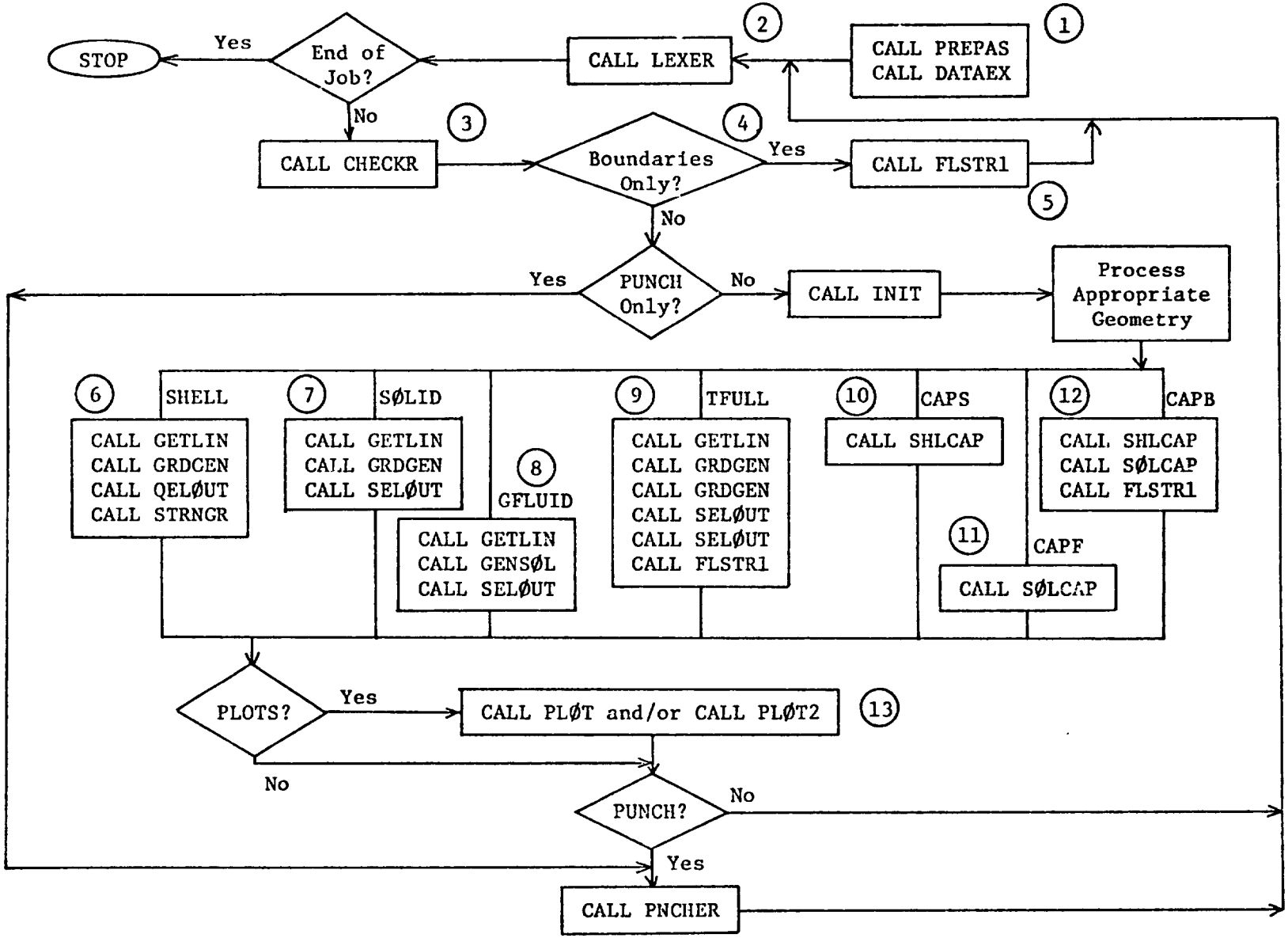


FIGURE 1. OVERALL FLOW OF MESHGEN

FIGURE 1. OVERALL FLOW OF MESHGEN, Cont'd

Description of Functions

(1) PREPAS and DATAEX are used to prepare the input MESHLAN sequence for processing.

(2) LEXER performs the lexical analysis of the MESHLAN sequence and builds much of the data base.

(3) CHECKR performs consistency checks on the input data.

(4) This test allows the option of generating a fluid and structure portion of the same model at different times. Both models may then be accessed from the SAVE file and fluid/structure interfaces defined.

(5) FLSTR1 generates the fluid/structure interface data.

(6) Sequence of subroutines required to generate SHELL models. GETLIN and GRDGEN generate grids, QELØUT elements, and STRNGR stringers.

(7) For the SØLID model, SELØUT generates elements.

(8) A general fluid (GFLUID) is defined by GENSØL and the associated elements are again found by SELØUT.

(9) TFULL (a combined SHELL and SØLID model) is simply the combination of the two portions, plus a call to FLSTR1 to generate fluid/structure boundaries.

(10) SHLCAP defines both the grid and element data for a shell cap.

(11) SØLCAP performs grid and element definitions for a solid cap.

(12) A shell cap containing fluid may be modeled with this routine.

(13) Two plot routines are available. PLØT generates three-dimensional plots, while PLØT2 generates two-dimensional pseudo-developed plots for some models.

Detailed descriptions of these routines and their associated utilities are described in subsequent sections.

2.1 FØRTRAN FILE DESCRIPTION

MESHGEN requires the use of six FØRTRAN files in addition to standard IØ units 5, 6 and 7. The file numbers are found in

CØMMØN/SYS2/IU1,IU2,IU3,IU4,IU5,IU6

These values are set by Block Data MESHBD and may be changed if desired. Operations on each of these files are described below.

File	Usage
IU1	Internal file used to store data passed to plot routines
IU2	Copy of input stream with MESHLAN data deleted
IU3	SAVE file for models (usually catalogued by user)
IU4	Scratch file used in computing fluid/structure interface
IU5	Available for defining external plot file for SC4020 plotter hardware
IU6	Copy of input stream after conversion to BCD

The actual formats of the data on each of these files is summarized below.

2.1.1 File IU1

Record	Contents
1	ITYPE - type of model: 1 = shell, 2 = solid
	NAME - 2-word BCD MØDEL name
	NR - number of radial stations
2	NT - number of circumferential stations
	NZ - number of axial stations
	ICLØSR - flag signifying radial closure
	ICLØST - flag signifying circumferential closure
3	NFLAG
	ANG1,ANG2 - plot view angles
	DØUT
4 thru n	GID - grid point ID
	R,T,Z - grid coordinates (cylindrical)

2.1.2 Files IU2 and IU6

The files are simply copies of the input stream, BCD, 80 characters per record.

2.1.3 File IU3

Record	Contents
1	NAME - 2-word BCD MØDEL name
2 thru n	Bulk Data images - 80 characters per record

In addition, the boundary element dimensions are saved to find boundaries at a later time. The format of such records is:

Word	Contents
1	BCD String '****'
2	Element ID
3,4	ZMIN and ZMAX for element
5,6	TMIN and TMAX for element

2.1.4 File IU4

This file is used internally to find the boundaries when all data is present, i.e., SHAPE=TFULL or CAPE. Its format is the same as that described above.

2.1.5 File IU5

The format of this file is determined by the SC4020 plot routines.

2.2 COMMON BLOCKS

2.2.1 CØMMØN/CØNTRL/MØDNAM(2),SHAPE,BØUND(3),NEW

MØDNAM - Array containing data from MØDEL command

1,2 MØDEL name, BCD

3 SAVE flag - 0 no SAVE
1 SAVE

SHAPE - Type of geometric shape being analyzed from SHAPE command

= 1 for SHELL
= 2 for SØLID
= 3 for TRUNCATED SØLID
= 4 for CONTAINED SØLID
= 5 for CAPSHELL
= 6 for CAPSØLID
= 7 for CAPBØTH

BØUND - Array storing boundary data from BØUNDARY command

1 - Type of boundary specification = -1 for INTERPØLATED TABLE
= 0 for FUNCTIØN
= 1 for TABLE

2 - Boundary ID, integer

3 - Fluid level for SHAPE=4 or 7

NEW - Flag to define status of SAVE file. Set to 1 if file is new.

2.2.2 CØMMØN/ELTAB/ELEM(10,8)

ELEM - Array holding element definition for up to 10 different elements per case. Column values are:

1 - Element type code
types 1-8 are quad plates
types 9-12 are fluids
types 13-17 are triangular plates

2 - Initial element ID, integer

3,4 - Not used

5 - Element property ID, integer

6 - Material orientation angle, α, real

7,8 - Used to denote THICKNESS VARIES command

2.2.3 CØMMØN/MLINE/NRL,NTL,NZL,R(200,9),T(200,9),Z(200,0)

This common block is used to store mesh line data generated by expanding and merging the zone definition data in /MESH/.

NRL,NTL,NZL - Number of mesh lines in each coordinate direction, integer

R,T,Z - Arrays holding expanded mesh line values. Each column of the array holds:

1 - Expanded coordinate values, real
2 - Expanded coordinate grid ID for shell integer
3 - Input coordinate system ID for shell integer
4 - Output coordinate system ID for shell integer
5 - Not used
6,7,8 - Same as 2,3,4 for the solid
9 - Not used

2.2.4 CØMMØN/MESH/G1,G2,ØPRØP,X1(5,20),X2(5,20),X3(5,20),NPT(3),STRING(6), SLØC(2,25)

G1 - Initial grid ID for shell, integer

G2 - Initial grid ID for solid, integer

ØPRØP - Overall property ID, integer.

X1,X2,X3 - Arrays storing zone definition values for each coordinate direction. The first subscript refers to the number of the STEP or DIVIDE command in the i^{th} direction, i.e., a maximum of 5 STEP or DIVIDE's may appear in each direction. The words within each entry are:

1. First value of coordinate on STEP command, real.
2. Second value of coordinate on STEP command, real.
3. Number of elements in the zone from STEP or DIVIDE command, integer.
4. Shell flag - set to 1 if plate elements are defined for the zone, integer.
5. Input coordinate system ID for the shell zone, INSYS, integer.
6. Output coordinate system ID for the shell zone, ØUTSYS, integer.
7. Grid point ID increment for shell zone from NUMBER command, integer.
8. Element ID increment for shell zone from NUMBER command, integer.
9. Pointer into ELEM array of /ELTAB/ defining the type of elements in the shell zone, integer.
10. Number of element types defined in the shell zone, integer.
11. Shell zone property ID from ZPRØP command, integer.
12. Material orientation angle, α, for shell zone from ZPRØP command, real.

13-19. Same as 4-10 for the solid zone.

20. Solid zone property ID from the ZPRØP command, integer.

NPT(I) - Number of zones defined in each coordinate direction.

STRING - 1 - Type of stringer, 0 for BAR, 1 for RØD.

2 - Stringer property ID, integer.

3 - Initial stringer ID number if θ direction, integer.

4 - Number of stringer stations in Z direction from ALØNG command, integer.

5 - Number of stringer stations in θ direction from ALØNG command, integer.

6 - Initial stringer ID number if Z direction, integer.

SLØC - Array containing stringer stations from ALØNG command in the Z and θ directions.

3.1 MAPFNS (MACHINE WORD FUNCTIONS)

3.1.1 Entry Points: LSHIFT, RSHIFT, ANDF, ØRF, XØRF, CØMPLF

3.1.2 Purpose

To perform basic computer word manipulations by standard binary digit (bit) operations. The manipulations are performed over the complete memory word length for the particular hardware.

3.1.3 Calling Sequence

All machine word functions are executed as FØRTRAN integer function subroutines with integer arguments.

3.1.4 Method

The method employed within each function will be described following the separate function examples.

3.1.5 Entries

K = LSHIFT (I,N)

The entire bit structure of word I is shifted left N places and the resulting word replaces word K. Word I is unchanged. High-order bits shifted out are lost. Zeros are supplied to vacated low-order positions. The shift is logical; no special provision is made for the sign position.

K = RSHIFT (I,N)

The entire bit structure of word I is shifted right N places and the resulting word replaces word K. Word I is unchanged. Low-order bits shifted out are lost. Zeros are supplied to vacated high-order positions. The shift is logical; no special provision is made for the sign position.

K = ANDF (I,J)

A logical product of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = ØRF (I,J)

A logical sum of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = XØRF (I,J)

The modulo-two sum (exclusive or) of the bits within word I and word J is formed and stored into word K. Words I and J are unchanged.

K = CØMPLF (I)

The ones complement of the bits within word I is formed and stored into word K. Word I is unchanged.

3.1.6 Design Requirements

NAPFNS is written in assembly language.

3.2 XRCARD (FREE-FIELD CARD DATA CONVERSION ROUTINE)

3.2.1 Entry Point: XRCARD

3.2.2 Purpose

To interpret free-field card input data as follows:

1. Identify BCD alpha and numeric data fields as they are converted and placed in the user's buffer.
2. Flag and output special data field delimiters.
3. Convert BCD numeric fields to binary integer or binary floating point.
4. Indicate when the data extends beyond one 72 column card.

3.2.3 Calling Sequence

```
CALL XRCARD(ØUTBUF,L,INBUF)
```

Where:

ØUTBUF = The buffer which is to contain the converted card image.

L = The length of ØUTBUF available to XRCARD.

INBUF = The buffer containing the card image to be converted.

3.2.4 Method

XRCARD's design is based on the necessity of having to function on a variety of computing machines having a variety of computer word structures, and a variety of differences in hollerith handling imposed by differing FØRTRAN compilers.

XRCARD analyzes the twenty hollerith words input through INBUF as follows:

Data Field Delimiters

Type A: The following symbols signify the end of an alpha field or numeric field on the card. As these symbols are encountered, they will be flagged and placed in the output buffer to aid the user in identifying the data.

(LEFT PAREN

/ SLASH

= EQUAL

* ASTERISK

Type B: The following symbols are identical to those listed above except that the symbol is not flagged or placed in the output buffer:

, COMMA

) RIGHT PAREN

When successive type A or type B delimiters are encountered, a null field indication (two BCD blank words) is output. A null field is generated for each successive delimiter. A null field is also generated when a type A or type B delimiter is followed by a $ indicating the end of data condition.

Type C: The following symbol is identical to the COMMA except that no null field indication is output when they are encountered in succession.

BLANK

End of Data Indication

There are three means by which end-of-data may be specified on the card:

- The last data field ends in column 72, or is followed by blanks out through column 72;
- $ is encountered, after which comments may be included out to column 80; or
- Continuation cards ending in (, /, = or , will result in a continuation flag (0 mode word)).

Format of Output Data

A mode word, N, is placed in the output buffer to distinguish between BCD data and numeric data.

Numeric Mode Word: A new mode word is output each time a numeric field is converted and output. (All numeric mode words are negative.)

N = -1 integer data (1 data word)
= -2 floating point single precision (1 data word)
= -4 floating point double precision (2 data words)

N indicates the type of numeric data and where to look for the next mode word.

Alpha Mode Word: When processing alpha data, only one mode word is output for successive alpha fields, i.e., an alpha mode word will never follow another alpha mode word.

N = The number of successive alpha fields encountered on the card. Each alpha field consists of two 4-character computer words (left adjusted). Thus, N can be used to compute the location of the next mode word.

The type A delimiters are output as alpha data and are 'covered' by the alpha mode word. Since data output in the alpha mode must consist of two words, a type A delimiter will appear as:

Word 1 = Delimiter flag, all bits of the word are on.
Word 2 = BCD delimiter, left adjusted, followed by BCD blanks.

End-of-Data: The end-of-data flag is placed last in the output buffer and appears in place of an expected mode word. There are two end-of-data flags:

- A word with all bits off, indicating that more data is to follow on a continuation card.
- A word will all bits on except for the sign, indicating that no more data is to follow for this card type.

3.2.5 Design Requirements

An alpha field must be eight characters or less. Long alpha fields will be truncated to eight characters.

All data must be placed in card columes 1-72.

A data field may not be split between two cards.

The specification of all numeric data fields must conform to FØRTRAN IV standards.

3.3 RE2AL (REAL NUMBER TO ALPHANUMERIC)

3.3.1 Entry Point: RE2AL

3.3.2 Purpose

To convert a single precision number to its BCD string representation.

3.3.3 Calling Sequence

```
CALL RE2AL(RE,ALPH)
```

where:

RE - single precision real number - real - input.

ALPH - BCD output (2 words)

3.3.4 Method

The output value is the floating or exponential form depending on which gives greater significant digits. Round off is based on the number of significant digits being output.

3.4 INT2AL

3.4.1 Entry Point: INT2AL

3.4.2 Purpose

To convert an integer value to its BCD string representation.

3.4.3 Calling Sequence

```
CALL INT2AL(NUMB,ALPH)
```

where:

NUMB - number to be converted to BCD string - integer - input

ALPH - two-word array containing BCD string - BCD - output

3.4.4 Method

Each digit of NUMB is isolated and the BCD string is built by appropriate shift and logical "OR" operations. If the integer value is greater than 8 digits, the string returned in ALPH is all "question marks" (?).

3.5 PAGE (PAGE HEADING)

3.5.1 Entry Point: PAGE

3.5.2 Purpose

To provide a standard page heading for MESHGEN output.

3.5.3 Calling Sequence

```
CALL PAGE
CØMMØN/SYSTEM/NLPP,NLINE,IPAGE
```

NLPP - Maximum number of lines per page - integer.

NLINE - Number of data lines on previous page - LINE is set to zero by PAGE.

IPAGE - Current page number - increased by 1 on each call to PAGE.

```
CØMMØN/ØUTPUT/NTITL(32),ITITL1(32),ITITL2(32),ITITL3(32)
```

3.5.4 Method

PAGE writes a standard 4 line heading from NTITL, ITITL1, ITITL2, ITITL3.

3.6 MESAGE (MESSAGE WRITER)

3.6.1 Entry Point: MESAGE

3.6.2 Purpose

To queue nonfatal messages during the execution of a module; and for fatal messages give a core dump (CALL PDUMP), print the message queue (CALL MSGWRT), and call PEXIT.

3.6.3 Calling Sequence

```
CALL MESAGE(ICØDE,L,M,N)
```

where:

ICØDE - Internal message number - integer - input.

L,M,N - Values to be included in the error message - mixed - input.

3.6.4 Method

Both fatal and nonfatal error messages are printed as they occur. If a fatal error occurs, the IERR flag is set and return is made to the calling routine where appropriate action must be taken.

3.7 BUG

3.7.1 Entry Point: BUG

3.7.2 Purpose

To allow for diagnostic output during program development.

3.7.3 Calling Sequence:

```
CALL BUG(IDENT,IDNUM,ARRAY,NWDS)
```

where:

IDENT - Four-character identifier to label output - BCD - input.

IDNUM - Number to identify output - integer - input.

ARRAY - Starting address of values to be output - input.

NWDS - Number of words to be output - integer - input.

```
CØMMØN/DIAG/IDIAG
```

3.7.4 Method

If the diagnostic flag IDIAG is on, the routine will print the specified data. The type of data and output formats are determined by the subroutine.

3.8 APRXEQ (APPROXIMATELY EQUAL FUNCTION)

3.8.1 Entry Point: APRXEQ

3.8.2 Purpose

To determine whether two single-precision real values are approximately equal within a given tolerance.

3.8.3 Calling Sequence

```
K = APRXEQ(A,B)  [LOGICAL FUNCTION]
```

where:

A,B - Numbers to be compared for approximate equality - real - input.

```
CØMMØN/SYS1/TØLER
```

where:

TØLER - Approximate equality tolerance - real.

3.8.4 Method

The following algorithm is used to determine approximate equality:

if A=B; then APRXEQ=.TRUE.
else

if $(A \neq 0 \wedge \left|\frac{A-B}{A}\right| < TØLER) \vee$

$(B \neq 0 \wedge \left|\frac{A-B}{B}\right| < TØLER)$; then APRXEQ=.TRUE.

else

APRXEQ=.TRUE.

3.9 DECØDE (DØF DECODER)

3.9.1 Entry Point: DECØDE

3.9.2 Purpose

To translate the normal NASTRAN degree-of-freedom code (string of integers 1-6 with no imbedded blanks) to a bit representation.

3.9.3 Calling Sequence

K = DECØDE(DØF) (INTEGER FUNCTION)

where:

DØF - NASTRAN degree-of-freedom code - integer - input.

3.9.4 Method

The digits of DØF are isolated and then a word is constructed where the i^{th} bit is on if the digit i appears in DØF. For example:

DØF = 126; DECØDE = 100011

DØF = 123456; DECØDE = 111111

DØF = 256; DECØDE = 110010

3.10 ENCØDE (DØF ENCODES)

3.10.1 Entry Point: INTEGER FUNCTIØN ENCØDE

3.10.2 Purpose

To translate a bit pattern to the NASTRAN degree-of-freedom code.

3.10.3 Calling Sequence

K = ENCØDE(DØF) (INTEGER FUNCTION)

where:

DØF - Bit string to be coded - integer - input.

3.10.4 Method:

Bits are extracted one at a time and the DØF string is generated. Examples of this process are:

011011 => 1245

111111 => 123456

000011 => 12

This routine is the inverse of DECØDE.

3.11 INTERP

3.11.1 Entry Point: INTERP

3.11.2 Purpose

To perform a linear interpolation on a tabular function r = f(z).

3.11.3 Calling Sequence

R = INTERP(Z) (REAL FUNCTION)

where

Z - value of axial coordinate at which radial value is desired - real - input.

COMMON/BTAB/

3.11.4 Method

A linear interpolation is performed to find the radial value. An error condition exists if $Z < Z_{min}$ or $Z > Z_{max}$

3.12 GETVAL

3.12.1 Entry Point: GETVAL

3.12.2 Purpose

To return a value of radial coordinate for a given axial value.

3.12.3 Calling Sequence

R = GETVAL(Z) [Real Function]

where:

Z - axial value; input, real

CØMMØN/CØNTRL/

3.12.4 Method

A test is made to determine whether the boundary has been specified by a FUNCTIØN or a TABLE. The appropriate call is then made to either FUNC or INTERP.

3.13 FUNC

3.13.1 Entry Point: FUNC

3.12.2 Purpose

Evaluates the incomplete quadratic function

$$a_4 z^2 + a_5 z + a_6 r^2 + a_7 r = a_8^2 + a_9$$

for r given a particular value of z.

3.12.3 Calling Sequence

R = FUNC(Z) (Real Function)

where:

Z - value of axial coordinate at which radial coordinate is desired; input, real.

COMMON/SHP/A(9)

A(I) - array containing quadratic coefficients; real.

3.13.4 Method

The following algorithm is used to find the value of r:

If $a_6 = a_7 = 0$ then ERROR

If $a_7 = 0$ then FUNC = $(a_8^2 + a_9 - a_4 z^2 - a_5 z)/a_7$

else DISC = $a_7^2 - 4a_6(a_4 z^2 + a_5 z - a_8^2 - a_9)$

if DISC 0.0 then ERROR

R1 = $(-a_7 + \sqrt{DISC})/2a_6$

R2 = $(-a_7 - \sqrt{DISC})/2a_6$

FUNC = max(R1,R2)

3.14 DIVLIN

3.14.1 Entry Point: DIVLIN

3.14.2 Purpose

To divide a boundary curve into n approximately equal segments.

3.14.3 Calling Sequence

```
CALL DIVLIN(NPT,ZP)
```

where:

NPT - the number of equally-spaced points - input - integer.

ZP - array of values at equally-spaced points (axial locations) - output - real.

```
COMMON/BTAB/    COMMON/SHP
COMMON/CONTRL/
```

3.14.4 Method

The SHAPE flag in the /CONTRL/ common block is checked to determine whether the boundary is defined by a function or table. In either event the length of the curve is approximated by the summation of line segments defining the curve. This length is then divided into NPT equal segments and interpolation is performed to give the axial values of these points.

3.15 PIF1

3.15.1 Entry Point: PIF1

3.15.2 Purpose

To perform a linear interpolation of the curve length function.

3.15.3 Calling Sequence

```
CALL PIF1(Z,L,N.LP,ZP)
```

where:

Z - table of axial values - real - input.

L - table of arc length at each Z - real - input.

N - number of points in tables - integer - input.

LP - particular value of length for which the axial value is desired - real - input.

ZP - interpolated value of Z at LP - real - output

4.1 FREØUT

4.1.1 Entry Point: FREØUT

4.1.2 Purpose

To generate fluid-free surface CFFREE Bulk Data cards.

4.1.3 Calling Sequence

```
CALL FREØUT (ID,IFACE)
```

where:

ID - Fluid element ID number - integer - input.

IFACE - Face identification for free surface - integer - input.

CØMMØN/LØADS/

4.1.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.2 GRDØUT

4.2.1 Entry Point: GRDØUT

4.2.2 Purpose

To generate GRID Bulk Data cards.

4.2.3 Calling Sequence

```
CALL GRDØUT(ID,IN,R,T,Z,ØUT,SPC)
```

where:

ID - grid point ID number - input - integer.

IN - input coordinate system ID - input - integer.

R,T,Z - grid point coordinates - input - real.

ØUT - output coordinate system ID - input - integer.

SPC - permanent single point constraint code - input - integer.

4.2.4 Method

Subroutine INT2AL is called for each input value and the Bulk Data card image is generated. The value SPC must be encoded back to NASTRAN form. In addition, if the PLØT flag is set, the grid ID and coordinates are written to the plot file. If the SAVE or PUNCH flags are on, appropriate action is taken.

4.3 HEXØUT

4.3.1 Entry Point: HEXØUT

4.3.2 Purpose

To generate CFHEX1 and CFHEX2 Bulk Data cards.

4.3.3 Calling Sequence

```
CALL HEXØUT(ITYPE,E,P,G1,G2,G3,G4,G5,G6,G7,G8)
```

where:

ITYPE - flag to determine type of card - input - integer.
= 9 CFHEX1
= 10 CFHEX2

E - element ID number - input - integer.

P - property ID number - input - integer.

G1-G8 - grid point ID number - input - integer.

4.3.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.4 PLØTEL

4.4.1 Entry Point: PLØTEL

4.4.2 Purpose

Generates PLØTEL Bulk Data cards for the free surface of a fluid model.

4.4.3 Calling Sequence

```
CALL PLØTEL(ID1,G1,G2,ID2,G3,G4)
```

where:

ID1,ID2 - PLØTEL element ID numbers - input - integer.

G1,G2,G3,G4 - grid point ID's defining the PLØTEL - input - integer.

4.4.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.5 PLTHED

4.5.1 Entry Point: PLTHED

4.5.2 Purpose

To write three header records on the scratch file used by the plotting routines.

4.5.3 Calling Sequence

```
CALL PLTHED(ITYPE,IPØS)
```

where:

ITYPE - type of plot that will be generated
= 1 for shell
= 2 for solid

IPØS - code for positioning of file
= 1 rewind
= 2 no rewind

4.5.4 Method

Twelve words are written as three four-word header records to the plot file. (See Section 2.1 for the format of this file.)

4.6 PNCHER

4.6.1 Entry Point: PNCHER

4.6.2 Purpose

To punch the Bulk Data for a model that resides on the SAVE file.

4.6.3 Calling Sequence

```
CALL PNCHER

COMMON/IODATA/JUNK(5),NAME(2)
```

where:

NAME - 2-word array containing the model name on the SAVE file.

4.6.4 Method

The SAVE file is accessed and a search is made for model NAME. If NAME is found, all Bulk Data images are punched. If NAME does not exist, the PUNCH command is ignored.

4.7 STRØUT

4.7.1 Entry Point: STRØUT

4.7.2 Purpose

To generate CRØD and CBAR Bulk Data cards.

4.7.3 Calling Sequence

```
CALL STRØUT(IC,EID,PID,G1,G2)
```

where:

IC - code for element type - input - integer
= 1 for CBAR
= 2 for CRØD

EID - element ID number - input - integer

PID - property ID number - input - integer

G1,G2 - grid point ID numbers - input - integer

4.7.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.8 TETØUT

4.8.1 Entry Point: TETØUT

4.8.2 Purpose

To generate CFTETRA Bulk Data cards.

4.8.3 Calling Sequence

```
CALL TETØUT(E,P,G1,G2,G3,G4)
```

where:

E - element ID number - input - integer.

P - property ID number - input - integer.

G1-G4 - grid point ID numbers - input - integer.

4.8.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.9 WEDØUT

4.9.1 Entry Point: WEDØUT

4.9.2 Purpose

To generate CFWEDGE Bulk Data cards.

4.9.3 Calling Sequence

```
CALL WEDØUT(E,P,G1,G2,G3,G4,G5,G6)
```

where:

E - element ID number - input - integer.

P - property ID number - input - integer.

G1-G6 - grid point ID numbers - input - integer.

4.9.4 Method

Subroutine INT2AL is called for each of the input values and the Bulk Data card image is generated. The image is then written to the output file. In addition, if the SAVE or PUNCH flags are on, the image will be written to the SAVE file and/or punched.

4.10 WRTHED

4.10.1 Entry Point: WRTHED

4.10.2 Purpose

To write a header record on the SAVE file if the SAVE option is elected for a given model.

4.10.3 Calling Sequence

```
CALL WRTHED

COMMON/CONTRL/
```

4.11 CCARD

4.11.1 Entry Point: CCARD

4.11.2 Purpose

To generate the Bulk Data card continuation field.

4.11.3 Calling Sequence

```
CALL CCARD(NUM,ØUT,PREFIX)
```

where:

NUM - numeric portion of continuation field - input - integer.

ØUT - 2-word array containing continuation field - output - BCD.

PREFIX - 2-character prefix for continuation field - input - BCD.

4.11.4 Method

Logical shifting and bitwise logical "OR" functions are performed to pack the prefix onto the BCD representation of NUM.

4.12 PREPAS

4.12.1 Entry Point: PREPAS

4.12.2 Purpose

To convert the input data stream from EBCDIC to BCD.

4.12.3 Calling Sequence

```
CALL PREPAS
```

4.12.4 Method

PREPAS performs a character-by-character conversion of the input stream and writes the input to unit IU6.

4.12.5 Design Requirements

PREPAS is an IBM-dependent routine and must be altered to simply copy the input data on other systems.

4.13 DATAEX

4.13.1 Entry Point: DATAEX

4.13.2 Purpose

To extract the MESHGEN functional data from the input stream in a pre-pass operation before mesh generation.

4.13.3 Calling Sequence

```
CALL DATAEX

CØMMØN/BØUND/
```

4.13.4 Method

The input file is read until the $DATA card is encountered. Each FUNCTIØN or TABLE definition is extracted and loaded into /BØUND/. When a hardware end-of-data is sensed on the input unit, a REWIND is performed and the MESHLAN sequence is copied to a scratch unit and all cards after and including $DATA are eliminated.

4.14 INIT

4.14.1 Entry Point: INIT

4.14.2 Purpose

To extract the boundary function or table data for the current case from the tables of all data input for the current job.

4.14.3 Calling Sequence

```
CALL INIT

CØMMØN/BØUND/FT(10,10),TT(500)
CØMMØN/SHP/A(9)
CØMMØN/BTAB/NPT,TABL(500)
```

/BØUND/ contains tables of all input functions and boundary tables

/SHP/ current function data

/BTAB/ current table data

4.14.4 Method

This routine extracts the current boundary data from the tables FT and TT. It then writes a summary of the data and copies it into A or TABL.

4.15 ZERØUT

4.15.1 Entry Point: ZERØUT

4.15.2 Purpose

To initialize the basic common blocks before executing each case in a MESHGEN run.

4.15.3 Calling Sequence

```
CALL ZERØUT

CØMMØN/MESH/          CØMMØN/SYSTEM/
CØMMØN/MLINE/         CØMMØN/IØDATA/
CØMMØN/CØNTRL/        CØMMØN/PSPC/
CØMMØN/ELTAB/
```

4.16 BLOCK DATA MESHBD

4.16.1 Entry Point: MESHBD

4.16.2 Purpose

To initialize machine-dependent parameters held in common blocks /SYS1/ and /SYS2/.

5. LEXICAL ANALYSIS SUBROUTINES

5.1 LEXER

5.1.1 Entry Point: LEXER

5.1.2 Purpose

To perform the lexical analysis of MESHLAN statements, and to initialize control arrays for each MØDEL.

5.1.3 Calling Sequence

```
CALL LEXER
```

5.1.4 Method

The logical flow of LEXER is shown in Figure 2. The routine determines the level of the MESHGEN command and calls the processing routine.

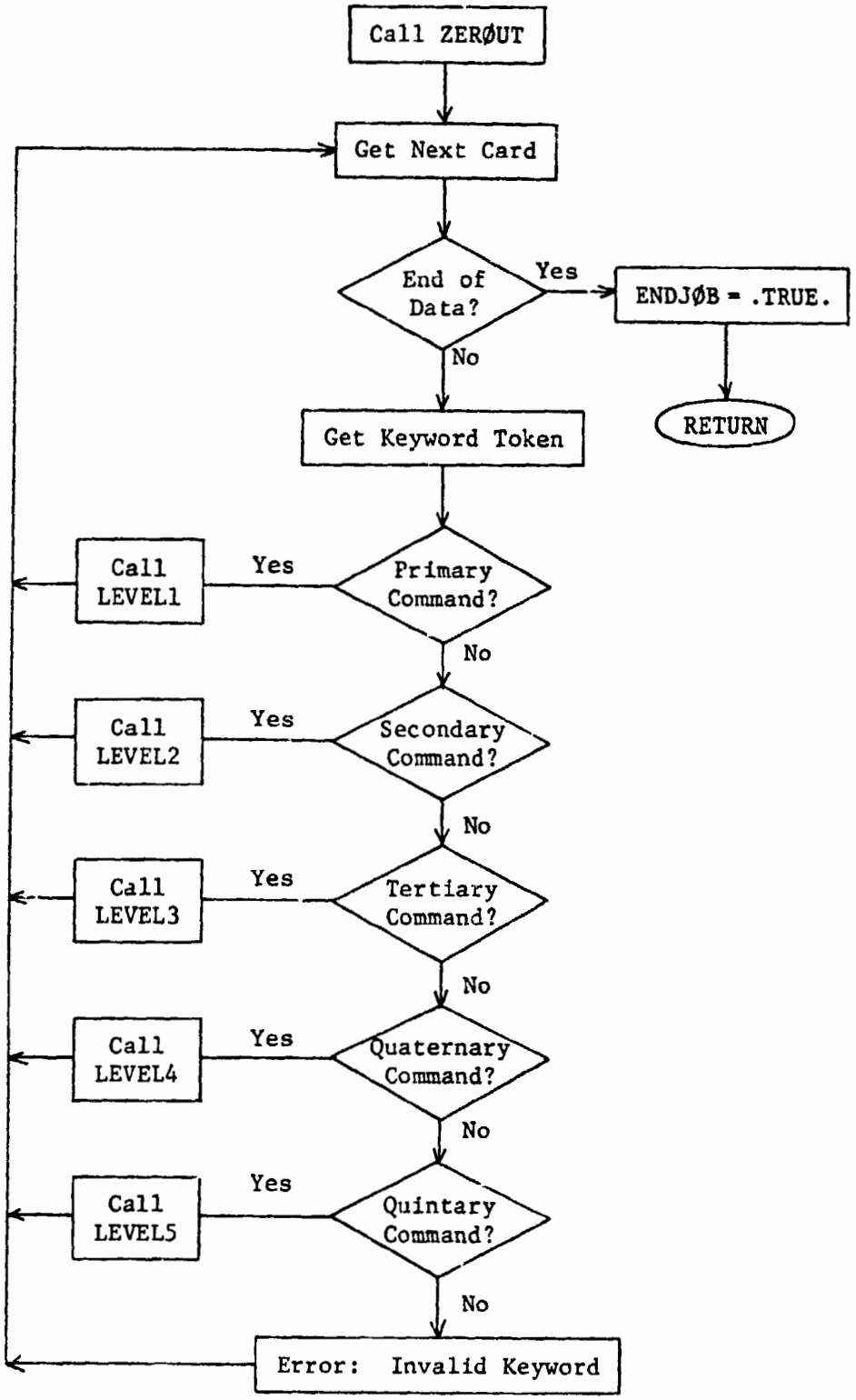


FIGURE 2. LOGICAL FLOW OF LEXER

5.2 LEVEL1

5.2.1 Entry Point: LEVEL1

5.2.2 Purpose

To translate and process the primary MESHLAN commands:

MØDEL, GEØMETRY, MESH, GRAV, PLØT, PLT2, PUNCH, ENDM, DIAG, FIND, MACH, and PLTH*.

5.2.3 Calling Sequence

```
CALL LEVEL1(P)
```

where:

P - Primary command ID - integer - input.

CØMMØN/MESH/	CØMMØN/MACHØN/
CØMMØN/LØADS/	CØMMØN/IØDATA/
CØMMØN/BFLSTR/	CØMMØN/CØNTRL/

5.2.4 Method

The primary commands are translated and values inserted into the appropriate common blocks.

*PLTHEAD is a facility-dependent command for MSFC.

5.3 LEVEL2

5.3.1 Entry Point: LEVEL2

5.3.2 Purpose

To translate and process the secondary MESHLAN commands:

SHAPE, BØUNDARY, STEP, DIVIDE, and ØPRØP.

5.3.3 Calling Sequence

CALL LEVEL2(P,S)

where:

P,S - Primary and secondary command ID's - input- integer.

CØMMØN/CØNTRL/

CØMMØN/MESH/

5.3.4 Method

Tests are performed to check the context of the secondary commands. The commands are then translated and the values inserted into the common blocks.

5.4 LEVEL3

5.4.1 Entry Point: LEVEL3

5.4.2 Purpose

To translate and process tertiary MESHLAN commands:

SHELL, SØLID, ZPRØP, and STRINGER.

5.4.3 Calling Sequence

```
CALL LEVEL3(P,S,T)
```

where:

P,S,T - Primary, secondary and tertiary command ID's - integer - input.

CØMMØN/MESH/

5.4.4 Method

Tests are performed to check the context of the tertiary commands. After translation, the appropriate values are inserted into /MESH/.

5.5 LEVEL4

5.5.1 Entry Point: LEVEL4

5.5.2 Purpose

To translate and process the quaternary MESHLAN commands:

NUMBER, INSYS, ØUTSYS, FIX, ELEMENTS, and ALØNG.

5.5.3 Calling Sequence

```
CALL LEVEL4(P,S,T,Q)
```

where:

P,S,T,Q - primary, secondary, etc. command ID's - input - integer.

CØMMØN/MESH/ CØMMØN/ELTAB/ CØMMØN/PSPC/

5.5.4 Method

Tests are performed to check the syntax of the LEVEL4 commands. The values extracted are then inserted into the appropriate common block.

5.6 LEVEL5

5.6.1 Entry Point: LEVEL5

5.6.2 Purpose

To process the quintary MESHLAN commands: PRØPERTY and THICKNESS.

5.6.3 Calling Sequence

```
CALL LEVEL5(P,S,T,Q1,Q2)
```

where:

P,S,T,Q1,Q2 - command ID for primary, secondary, etc. command sequence.

CØMMØN/ELTAB/ELEM(10,8)

5.6.4 Method

Tests are performed to check the syntax of the LEVEL5 commands. Extracted values are then placed in the /ELTAB/ common block.

5.7 CHECKR

5.7.1 Entry Point: CHECKR

5.7.2 Purpose

To test for the allowability and consistency of the processed MESHLAN sequence.

5.7.3 Calling Sequence

```
CALL CHECKR

CØMMØN/MESH/          CØMMØN/ELTAB/
CØMMØN/LØADS/         CØMMØN/CØNTRL/
CØMMØN/BFLSTR/
```

5.8 GETTØK

5.8.1 Entry Point: GETTØK

5.8.2 Purpose

To return the next token of a MESHLAN command and its type.

5.8.3 Calling Sequence

```
CALL GETTØK

COMMON/LEXIC/TCØDE,TØK(2)
```

where:

TCØDE - is the token type (see Section 3.2)
TØK - is the actual value of the token

5.9 GETCRD

5.9.1 Entry Point: GETCRD

5.9.2 Purpose

To process the MESHLAN commands for lexical analysis.

5.9.3 Calling Sequence

```
CALL GETCRD

CØMMØN/CARD/TØKEN(3,100),NTØK
```

where:

TØK - Array containing the types and values of each token extracted from the input stream. The contents are:

Row	Contents
1	Token type code 1 - BCD 2 - Integer 3 - Real 4 - Delimiter 5 - End-of-card
2	Value of token

5.9.4 Method

When GETCRD is called, it reads the next card image in the input stream. XRCARD is then called to extract the tokens from the card. This routine then processes the XRCARD output to a simpler form to be used by the lexical routines. Although this function is purely overhead, it is of great use in simplifying coding and understanding of the lexical routines LEVEL1 - LEVEL5.

6. PLOTTING SUBROUTINES

6.1 PLØT2

6.1.1 Entry Points: PLØT, PLØT2

6.1.2 Purpose

To drive the various plotting subroutines.

6.1.3 Calling Sequence

CALL PLØT2(FRAME) - 2-D plots

CALL PLØT(FRAME) - 3-D plots

where:

FRAME - last frame number, zero initially

6.1.4 Method

The two entry points set a flag, IPLT23, that determines if 2-D or 3-D plots are to be made. The title is printed, then PLTRD is called to initialize the on-core arrays. If the plot can be made, various subroutines are called to do each aspect.

6.2 PLTNFR

6.2.1 Entry Point: PLTNFR

6.2.2 Purpose

For SC-4020 plots, skip to a new frame, write the plot titles, and increment the frame number.

6.2.3 Calling Sequence

```
CALL PLTNFR(FRAME)
```

where:

FRAME - input/output frame number

6.2.4 Method

If the frame number is ≤ 0, the SC-4020 is initialized. The plot titles are generated from data in /TITLE/ and /PLTDAT/.

6.2.5 Additional Subroutines

SC-4020 routines LABLV, CAMAV, FRAMEV, and PRINTV are called.

6.3 PLTRD

6.3.1 Entry Point: PLTRD

6.3.2 Purpose

For SC-4020 plots, create the in-core arrays of grid data. The data is rotated as per user request and converted to rasters. Each type of element input causes a resorting and/or addition of points on the input file.

6.3.3 Calling Sequence

```
CALL PLTRD(IFIL,IGRL,IGR,RGR)
```

where:

IFIL - input file number, binary records of 4 words.

IGRL - input, number of words of open core.

IGR, RGR - output array of grid data, 3 words per point

word 1 = grid ID
word 2 = X raster
word 3 = Y raster

6.3.4 Method

Each of the element types, defined in the header record of the input file, causes a special sort of the grid data output. For closure in Z or theta, extra points are introduced.

6.4 TPLAB

6.4.1 Entry Point: TPLAB

6.4.2 Purpose

For SC-4020 plots, places the grid identification number to the right of the grid point.

6.4.3 Calling Sequence

```
CALL TPLAB(IGR)
```

where:

IGR - 3-word array of grid data

Word	Description
1	grid ID number
2	X raster
3	Y raster

6.4.4 Method

The grid set is converted to alphanumeric and printed 32 rasters to the right of the point.

6.4.5 Additional Subroutines

SC-4020 routine PRINTV is required.

6.5 TPLIN

6.5.1 Entry Point: TPLIN

6.5.2 Purpose

For SC-4020 plots, to call one of the four line-drawing routines.

6.5.3 Calling Sequence

```
CALL TPLIN(IGR)
```

where:

IGR - input 3-word grid ID and raster array

6.6 TYPL1

6.6.1 Entry Point: TYPL1

6.6.2 Purpose

For SC-4020 plots, to draw the elements for a shell of revolution model.

6.6.3 Calling Sequence

```
CALL TYPL1(IGR)
```

where:

IGR - 3-word grid ID and raster array

6.6.4 Method

The pre-defined array of IGR set up by PLTRD is scanned to extract raster locations.

6.6.5 Additional Subroutines

SC-4020 routine LINEV is required.

6.7 TYPL2

6.7.1 Entry Point: TYPL2

6.7.2 Purpose

For SC-4020 plots, to draw the elements for a solid of revolution.

6.7.3 Calling Sequence

```
CALL TYPL2(IGR)
```

where:

IGR - 3-word grid ID and raster array

6.7.4 Method

The pre-defined array, IGR, set up by PLTRD is scanned to extract raster locations.

6.7.5 Additional Subroutines

SC-4020 routine LINEV is required.

6.8 TYPL3

6.8.1 Entry Point: TYPL3

6.8.2 Purpose

For SC-4020 plots, to draw the elements for a spherical shell cap.

6.8.3 Calling Sequence

```
CALL TYPL3(IGR)
```

where:

IGR - 3-word grid ID and raster array

6.8.4 Method

The grid-defined array, IGR, set up by PLTRD is scanned to extract raster locations.

6.8.5 Additional Subroutines

SC-4020 routine LINEV is required.

6.9 TYPL4

6.9.1 Entry Point: TYPL4

6.9.2 Purpose

For SC-4020 plots, to draw the elements for a spherical solid cap.

6.9.3 Calling Sequence

```
CALL TYPL4(IGR)
```

where:

IGR - 3-word grid ID and raster array

6.9.4 Method

The pre-defined array, IGR, set up by PLTRD is called to extract raster locations.

6.9.5 Additional Subroutines

SC-4020 routine LINEV is required.

7. COMPUTATIONAL SUBROUTINES

7.1 GETLIN

7.1.1 Entry Point: GETLIN

7.1.2 Purpose

To generate the mesh lines in each coordinate direction and apply permanent single-point constraints.

7.1.3 Calling Sequence

```
CALL GETLIN
```

CØMMØN/CØNTRL/	CØMMØN/PSPC/
CØMMØN/MLINE/	CØMMØN/SHP/
CØMMØN/MESH/	CØMMØN/BTAB/

7.1.4 Method

The mesh definition data is accessed from /MESH/. This data is then expanded to define all the coordinate values for the model. These are then stored in /MLINE/ (see Section 2.2.3). In addition to this expansion, SPCs are also applied to each grid line.

7.2 QELØUT

7.2.1 Entry Point: QELØUT

7.2.2 Purpose

To generate element connection for shell models.

7.2.3 Calling Sequence

```
CALL QELØUT

CØMMØN/CØNTRL/          CØMMØN/MLINE/
CØMMØN/MESH/            CØMMØN/ELTAB/
```

7.3 SELØUT

7.3.1 Entry Point: SELØUT

7.3.2 Purpose

To generate element data for solids of revolution (SØLID, TFULL, GFLUID)

7.3.3 Calling Sequence

```
CALL SELØUT

CØMMØN/MESH/          CØMMØN/CØNTRL/
CØMMØN/MLINE/         CØMMØN/ELTAB/
```

7.3.4 Method

The grid point ID numbers and increments are accessed and the appropriate solid element connection Bulk Data is generated. I. addition, PLØTEL data is generated for free surfaces and element ranges computed for CFLSTR Bulk Data.

7.4 GENSØL

7.4.1 Entry Point: GENSØL

7.4.2 Purpose

To generate grid points for the generalized solid model (GFLUID).

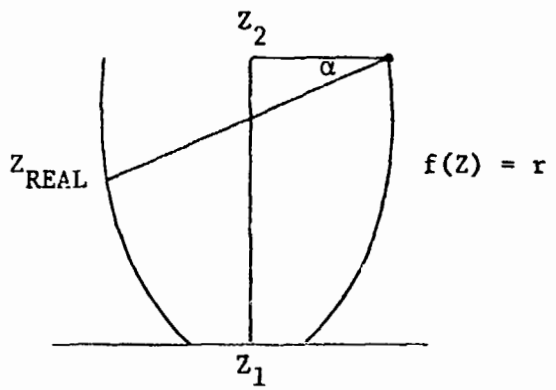
7.4.3 Calling Sequence

```
CALL GENSØL

CØMMØN/CØNTRL/          CØMMØN/MLINE/
CØMMØN/BTAB/            CØMMØN/SHP/
CØMMØN/MESH/
```

7.4.4 Method

GENSØL must compute all coordinates for a fluid with a tilted surface knowing only the boundary function and the tilt angle. The first step is to compute the height (axial value) of the point on the low end of the surface as shown in the following illustration:



The equation of the surface line is defined by:

$$m = \tan\left(\frac{\Pi\alpha}{180}\right) \qquad (1)$$

$$Z = mr + Z_1 - mf(Z_1) \qquad (2)$$

The value Z_{REAL} is obtained by solving this linear equation simultaneously with r = f(Z). Once these values are found, coordinates are generated at each elevation, circumferential and radial station by interpolation and solution of Eq. (2). At each step, GRID Bulk Data is generated. A subsequent call to SELØUT generates the element connection cards.

7.5 SHLCAP

7.5.1 Entry Point: SHLCAP

7.5.2 Purpose

To generate both grid points and element connections for the shell cap model.

7.5.3 Calling Sequence

```
CALL SHLCAP

CØMMØN/CØNTRL/          CØMMØN/ELTAB/
CØMMØN/MESH/            CØMMØN/PSDC/
```

7.6 SØLCAP

7.6.1 Entry Point: SØLCAP

7.6.2 Purpose

To generate both grid points and element connections for the solid cap model. In addition, to generate PLØTELs for the fluid free surface, if applicable.

7.6.3 Calling Sequence

```
CALL SØLCAP

CØMMØN/MLINE/          CØMMØN/CØNTRL/
CØMMØN/BTAB/           CØMMØN/MESH/
CØMMØN/SHP/            CØMMØN/ELEM/
```

7.7 STRNGR

7.7.1 Entry Point: STRNGR

7.7.2 Purpose

To generate stringer connections (CBAR or CRØD) defined with shell models.

7.7.3 Calling Sequence

```
CALL STRNGR

CØMMØN/MESH/
CØMMØN/MLINE/
```

7.8 FLSTR1

7.8.1 Entry Point: FLSTR1

7.8.2 Purpose

To determine the fluid/structure interfaces and generate CFLSTR Bulk Data cards.

7.8.3 Calling Sequence

```
CALL FLSTR1(ICØDE)
```

where:

ICØDE - specifies whether the shell elements are on the SAVE file (IU3) or the scratch file (IU4)

7.8.4 Method

The structure boundary ranges (TMIN, TMAX, ZMIN, ZMAX; see Section 2.1.3) are read from IU3 or IU4 into core. The fluid element ranges are then read from the SAVE file one at a time and compared with the shell ranges. All overlapping shell element ID's are saved. When the process is completed for a given fluid element, a CFLSTR Bulk Data card (or cards) is generated and the next fluid element is read. This process continues until all fluid elements have been tested.

8. SYSTEM CONSIDERATIONS

MESHGEN was written for, and implemented on, an IBM 360/165. It is currently being implemented on the UNIVAC 1108. With the overlay structure defined in Figure 3, MESHGEN will execute in 180K decimal bytes (45K words). This number can be affected by local modifications to system routines, FORTRAN IØ packages, or SC4020 software.

A list of subroutines that can cause implementation problems when installing on a new system are summarized.

8.1 OVERLAY FOR MESHGEN

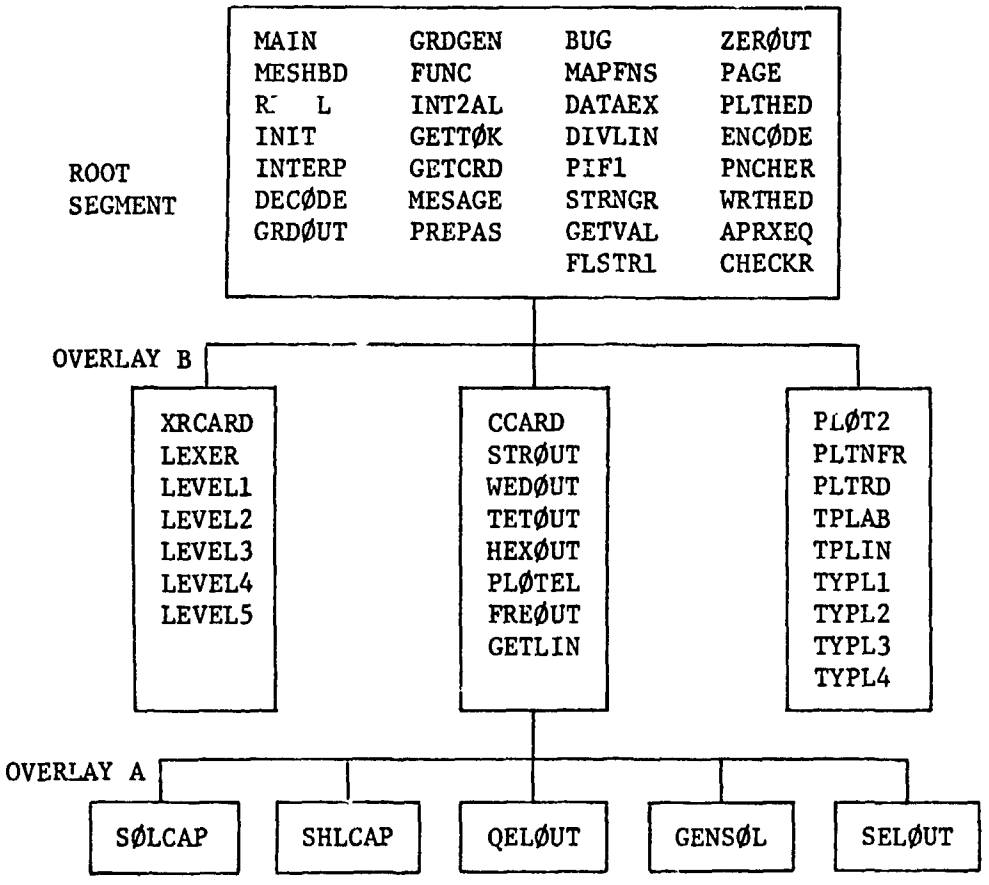


FIGURE 3. OVERLAY FOR MESHGEN

8.2 MACHINE OR FACILITY DEPENDENT ROUTINES

8.2.1 MAPFNS

This routine is written in assembly language and must be written for each system before implementation.

8.2.2 PREPAS

Used to convert EBCDIC to BCD before processing begins. For systems utilizing BCD only, the routine should be rewritten to simply copy the input.

8.2.3 MESHBD

The machine constants defined in this BLOCK DATA should be changed to reflect the system architecture.

8.2.4 PLØTS

The plot driver routine, PLØT2, as well as all plot interfaces should be checked to conform to facility-dependent conversion.